

MuPDFCore

2.0.0

Generated by Doxygen 1.9.5

1 MuPDFCore: Multiplatform .NET bindings for MuPDF	1
1.1 Getting started	1
1.2 Usage	2
1.2.1 Documentation	2
1.2.2 Minimal working example	2
1.2.3 Examples	2
1.3 Building from source	3
1.3.1 1. Building libmupdf	3
1.3.2 2. Building MuPDFWrapper	4
1.3.2.1 Windows (x86 and x64)	4
1.3.2.2 Windows (arm64)	5
1.3.2.3 macOS and Linux	5
1.3.3 3. Creating the native assets MuPDFCore NuGet packages	5
1.3.4 4. Creating the MuPDFCore NuGet package	6
1.3.5 5. Running tests	6
1.3.5.1 Windows	7
1.3.5.2 macOS and Linux	7
1.4 Note about MuPDFCore and .NET Framework 	7
2 Namespace Index	9
2.1 Package List	9
3 Hierarchical Index	11
3.1 Class Hierarchy	11
4 Class Index	13
4.1 Class List	13
5 File Index	17
5.1 File List	17
6 Namespace Documentation	19
6.1 Avalonia Namespace Reference	19
6.2 Avalonia.Animation Namespace Reference	19
6.3 MuPDFCore Namespace Reference	19
6.3.1 Enumeration Type Documentation	22
6.3.1.1 BoxType	22
6.3.1.2 ColorSpaceType	22
6.3.1.3 DocumentOutputFileTypes	22
6.3.1.4 DocumentRestrictions	22
6.3.1.5 EncryptionState	23
6.3.1.6 ExitCodes	23
6.3.1.7 InputFileTypes	23
6.3.1.8 PasswordTypes	23

6.3.1.9 PixelFormats	23
6.3.1.10 RasterOutputFileTypes	24
6.3.1.11 RestrictionState	24
6.4 MuPDFCore.MuPDFRenderer Namespace Reference	24
7 Class Documentation	25
7.1 MuPDFCore.BGRMuPDFColorSpace Class Reference	25
7.1.1 Detailed Description	25
7.1.2 Property Documentation	26
7.1.2.1 NumBytes	26
7.1.2.2 Type	26
7.2 MuPDFCore.CBZCreationOptions Class Reference	26
7.2.1 Detailed Description	27
7.2.2 Member Enumeration Documentation	27
7.2.2.1 ColorSpace	27
7.2.2.2 RasterizerOption	27
7.2.3 Property Documentation	27
7.2.3.1 Alpha	27
7.2.3.2 GraphicsRasterizer	28
7.2.3.3 Height	28
7.2.3.4 IncludeAnnotations	28
7.2.3.5 RenderingColorSpace	28
7.2.3.6 Rotate	28
7.2.3.7 TextRasterizer	29
7.2.3.8 Width	29
7.2.3.9 XResolution	29
7.2.3.10 YResolution	29
7.3 MuPDFCore.CMYKMuPDFColorSpace Class Reference	30
7.3.1 Detailed Description	30
7.3.2 Property Documentation	30
7.3.2.1 NumBytes	30
7.3.2.2 Type	31
7.4 MuPDFCore.MuPDFDocument.Create Class Reference	31
7.4.1 Detailed Description	33
7.4.2 Member Function Documentation	33
7.4.2.1 CBZDocument() [1/4]	33
7.4.2.2 CBZDocument() [2/4]	33
7.4.2.3 CBZDocument() [3/4]	34
7.4.2.4 CBZDocument() [4/4]	34
7.4.2.5 Document() [1/4]	34
7.4.2.6 Document() [2/4]	35
7.4.2.7 Document() [3/4]	35

7.4.2.8 Document() [4/4]	36
7.4.2.9 HTMLDocument() [1/4]	36
7.4.2.10 HTMLDocument() [2/4]	36
7.4.2.11 HTMLDocument() [3/4]	38
7.4.2.12 HTMLDocument() [4/4]	38
7.4.2.13 PDFDocument() [1/4]	39
7.4.2.14 PDFDocument() [2/4]	39
7.4.2.15 PDFDocument() [3/4]	39
7.4.2.16 PDFDocument() [4/4]	40
7.4.2.17 StructuredTextDocument() [1/4]	40
7.4.2.18 StructuredTextDocument() [2/4]	40
7.4.2.19 StructuredTextDocument() [3/4]	41
7.4.2.20 StructuredTextDocument() [4/4]	41
7.4.2.21 SVGDocument() [1/2]	42
7.4.2.22 SVGDocument() [2/2]	42
7.4.2.23 TextDocument() [1/4]	42
7.4.2.24 TextDocument() [2/4]	43
7.4.2.25 TextDocument() [3/4]	43
7.4.2.26 TextDocument() [4/4]	44
7.4.2.27 XHTMLDocument() [1/4]	44
7.4.2.28 XHTMLDocument() [2/4]	44
7.4.2.29 XHTMLDocument() [3/4]	45
7.4.2.30 XHTMLDocument() [4/4]	45
7.5 MuPDFCore.DisposableIntPtr Class Reference	46
7.5.1 Detailed Description	46
7.5.2 Constructor & Destructor Documentation	46
7.5.2.1 DisposableIntPtr() [1/2]	46
7.5.2.2 DisposableIntPtr() [2/2]	47
7.5.3 Member Function Documentation	47
7.5.3.1 Dispose()	47
7.6 MuPDFCore.DocumentLockedException Class Reference	47
7.6.1 Detailed Description	48
7.7 MuPDFCore.PDFCreationOptions.DocumentPermissions Class Reference	48
7.7.1 Detailed Description	49
7.7.2 Member Enumeration Documentation	49
7.7.2.1 ExtractionPermission	49
7.7.2.2 FormPermission	49
7.7.2.3 PrintingPermission	49
7.7.3 Property Documentation	50
7.7.3.1 AllowDocumentModification	50
7.7.3.2 AllowDocumentRestructuring	50
7.7.3.3 Annotations	50

7.7.3.4 Extraction	50
7.7.3.5 Printing	51
7.8 MuPDFCore.GrayscaleMuPDFColorSpace Class Reference	51
7.8.1 Detailed Description	51
7.8.2 Property Documentation	52
7.8.2.1 NumBytes	52
7.8.2.2 Type	52
7.9 MuPDFCore.HTMLCreationOptions Class Reference	52
7.9.1 Detailed Description	53
7.9.2 Property Documentation	53
7.9.2.1 Dehyphenate	53
7.9.2.2 IncludeAnnotations	53
7.9.2.3 IncludeCharactersOutsideMediaBox	53
7.9.2.4 InhibitSpaces	53
7.9.2.5 PreserveImages	54
7.9.2.6 PreserveLigatures	54
7.9.2.7 PreserveSpans	54
7.9.2.8 PreserveWhitespace	54
7.9.2.9 UseCIDForUnknownUnicode	54
7.10 MuPDFCore.IndexedMuPDFColorSpace Class Reference	55
7.10.1 Detailed Description	55
7.10.2 Property Documentation	55
7.10.2.1 BaseColorSpace	55
7.10.2.2 LookupTable	56
7.10.2.3 NumBytes	56
7.10.2.4 Type	56
7.11 MuPDFCore.LabMuPDFColorSpace Class Reference	56
7.11.1 Detailed Description	57
7.11.2 Property Documentation	57
7.11.2.1 NumBytes	57
7.11.2.2 Type	57
7.12 MuPDFCore.LifetimeManagementException< T1, T2 > Class Template Reference	57
7.12.1 Detailed Description	58
7.12.2 Constructor & Destructor Documentation	58
7.12.2.1 LifetimeManagementException()	58
7.12.3 Property Documentation	58
7.12.3.1 Disposing	59
7.12.3.2 Message	59
7.12.3.3 Owner	59
7.13 MuPDFCore.MessageEventArgs Class Reference	59
7.13.1 Detailed Description	60
7.13.2 Constructor & Destructor Documentation	60

7.13.2.1 MessageEventArgs()	60
7.13.3 Property Documentation	60
7.13.3.1 Message	60
7.14 MuPDFCore.MuPDF Class Reference	61
7.14.1 Detailed Description	61
7.14.2 Member Function Documentation	61
7.14.2.1 RedirectOutput()	61
7.14.2.2 ResetOutput()	62
7.14.3 Event Documentation	62
7.14.3.1 StandardErrorMessage	62
7.14.3.2 StandardOutputMessage	62
7.15 MuPDFCore.MuPDFColorSpace Class Reference	63
7.15.1 Detailed Description	63
7.15.2 Property Documentation	64
7.15.2.1 Name	64
7.15.2.2 NumBytes	64
7.15.2.3 RootColorSpace	64
7.15.2.4 Type	64
7.16 MuPDFCore.MuPDFContext Class Reference	65
7.16.1 Detailed Description	65
7.16.2 Constructor & Destructor Documentation	66
7.16.2.1 MuPDFContext()	66
7.16.3 Member Function Documentation	66
7.16.3.1 ClearStore()	66
7.16.3.2 Dispose()	66
7.16.3.3 ShrinkStore()	66
7.16.4 Property Documentation	67
7.16.4.1 AntiAliasing	67
7.16.4.2 GraphicsAntiAliasing	67
7.16.4.3 StoreMaxSize	67
7.16.4.4 StoreSize	67
7.16.4.5 TextAntiAliasing	68
7.17 MuPDFCore.MuPDFDocument Class Reference	68
7.17.1 Detailed Description	71
7.17.2 Constructor & Destructor Documentation	71
7.17.2.1 MuPDFDocument() [1/5]	71
7.17.2.2 MuPDFDocument() [2/5]	72
7.17.2.3 MuPDFDocument() [3/5]	72
7.17.2.4 MuPDFDocument() [4/5]	73
7.17.2.5 MuPDFDocument() [5/5]	73
7.17.3 Member Function Documentation	73
7.17.3.1 ClearCache()	73

7.17.3.2 CreateDocument() [1/2]	74
7.17.3.3 CreateDocument() [2/2]	74
7.17.3.4 Dispose()	74
7.17.3.5 ExtractText() [1/2]	75
7.17.3.6 ExtractText() [2/2]	75
7.17.3.7 ExtractTextAsync()	76
7.17.3.8 GetMultiThreadedRenderer()	76
7.17.3.9 GetRenderedSize() [1/2]	77
7.17.3.10 GetRenderedSize() [2/2]	77
7.17.3.11 GetStructuredTextPage() [1/2]	78
7.17.3.12 GetStructuredTextPage() [2/2]	78
7.17.3.13 GetStructuredTextPageAsync()	80
7.17.3.14 Layout()	81
7.17.3.15 LayoutSinglePage()	81
7.17.3.16 Render() [1/6]	81
7.17.3.17 Render() [2/6]	82
7.17.3.18 Render() [3/6]	82
7.17.3.19 Render() [4/6]	84
7.17.3.20 Render() [5/6]	84
7.17.3.21 Render() [6/6]	85
7.17.3.22 SaveImage() [1/2]	86
7.17.3.23 SaveImage() [2/2]	86
7.17.3.24 SaveImageAsJPEG() [1/2]	87
7.17.3.25 SaveImageAsJPEG() [2/2]	87
7.17.3.26 TryUnlock() [1/2]	88
7.17.3.27 TryUnlock() [2/2]	88
7.17.3.28 WriteImage() [1/2]	89
7.17.3.29 WriteImage() [2/2]	89
7.17.3.30 WriteImageAsJPEG() [1/2]	90
7.17.3.31 WriteImageAsJPEG() [2/2]	90
7.17.4 Property Documentation	91
7.17.4.1 ClipToPageBounds	91
7.17.4.2 EncryptionState	91
7.17.4.3 OptionalContentGroupData	91
7.17.4.4 Outline	91
7.17.4.5 Pages	92
7.17.4.6 Restrictions	92
7.17.4.7 RestrictionState	92
7.18 MuPDFCore.MuPDFException Class Reference	92
7.18.1 Detailed Description	93
7.18.2 Member Data Documentation	93
7.18.2.1 ErrorCode	93

7.19 MuPDFCore.MuPDFExternalLinkDestination Class Reference	93
7.19.1 Detailed Description	94
7.19.2 Property Documentation	94
7.19.2.1 Type	94
7.19.2.2 Uri	94
7.20 MuPDFCore.MuPDFFont Class Reference	95
7.20.1 Detailed Description	95
7.20.2 Member Function Documentation	96
7.20.2.1 Dispose()	96
7.20.2.2 GetFreeTypeHandle()	96
7.20.2.3 GetType3Handle()	96
7.20.3 Property Documentation	97
7.20.3.1 IsBold	97
7.20.3.2 IsItalic	97
7.20.3.3 IsMonospaced	97
7.20.3.4 IsSerif	97
7.20.3.5 Name	98
7.21 MuPDFCore.MuPDFImage Class Reference	98
7.21.1 Detailed Description	99
7.21.2 Member Enumeration Documentation	99
7.21.2.1 ImageOrientation	99
7.21.3 Member Function Documentation	99
7.21.3.1 Dispose()	99
7.21.3.2 GetBytes() [1/2]	99
7.21.3.3 GetBytes() [2/2]	100
7.21.3.4 Save()	100
7.21.3.5 SaveAsJPEG()	101
7.21.3.6 Write()	101
7.21.3.7 WriteAsJPEG()	102
7.21.4 Property Documentation	102
7.21.4.1 ColorSpace	102
7.21.4.2 Height	103
7.21.4.3 Orientation	103
7.21.4.4 ParentBlock	103
7.21.4.5 Width	103
7.21.4.6 XRes	103
7.21.4.7 YRes	104
7.22 MuPDFCore.MuPDFInternalLinkDestination Class Reference	104
7.22.1 Detailed Description	105
7.22.2 Member Enumeration Documentation	105
7.22.2.1 InternalDestinationType	105
7.22.3 Property Documentation	105

7.22.3.1 Chapter	105
7.22.3.2 Height	106
7.22.3.3 InternalType	106
7.22.3.4 Page	106
7.22.3.5 PageNumber	106
7.22.3.6 Type	106
7.22.3.7 Width	107
7.22.3.8 X	107
7.22.3.9 Y	107
7.22.3.10 Zoom	107
7.23 MuPDFCore.MuPDFLink Class Reference	107
7.23.1 Detailed Description	108
7.23.2 Property Documentation	108
7.23.2.1 ActiveArea	108
7.23.2.2 Destination	108
7.23.2.3 IsVisible	108
7.24 MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs Class Reference	109
7.24.1 Detailed Description	109
7.24.2 Constructor & Destructor Documentation	109
7.24.2.1 MupdfLinkClickedEventArgs()	109
7.24.3 Property Documentation	110
7.24.3.1 LinkDestination	110
7.25 MuPDFCore.MuPDFLinkDestination Class Reference	110
7.25.1 Detailed Description	111
7.25.2 Member Enumeration Documentation	111
7.25.2.1 DestinationType	111
7.25.3 Property Documentation	111
7.25.3.1 Type	111
7.26 MuPDFCore.MuPDFLinks Class Reference	112
7.26.1 Detailed Description	112
7.26.2 Member Function Documentation	112
7.26.2.1 Dispose()	112
7.26.2.2 GetEnumerator()	113
7.26.3 Property Documentation	113
7.26.3.1 Count	113
7.26.3.2 this[int index]	113
7.27 MuPDFCore.MuPDFMultiThreadedPageRenderer Class Reference	113
7.27.1 Detailed Description	114
7.27.2 Member Function Documentation	114
7.27.2.1 Abort()	114
7.27.2.2 Dispose()	115
7.27.2.3 GetProgress()	115

7.27.2.4 GetSpanItem()	115
7.27.2.5 Render() [1/2]	115
7.27.2.6 Render() [2/2]	116
7.27.3 Property Documentation	116
7.27.3.1 ThreadCount	117
7.28 MuPDFCore.MuPDFOptionalContentGroup Class Reference	117
7.28.1 Detailed Description	117
7.28.2 Property Documentation	117
7.28.2.1 IsEnabled	117
7.28.2.2 Name	118
7.29 MuPDFCore.MuPDFOptionalContentGroupCheckbox Class Reference	118
7.29.1 Detailed Description	118
7.30 MuPDFCore.MuPDFOptionalContentGroupConfiguration Class Reference	119
7.30.1 Detailed Description	119
7.30.2 Member Function Documentation	119
7.30.2.1 Activate()	119
7.30.3 Property Documentation	119
7.30.3.1 Creator	120
7.30.3.2 IsDefault	120
7.30.3.3 Name	120
7.30.3.4 UI	120
7.31 MuPDFCore.MuPDFOptionalContentGroupData Class Reference	120
7.31.1 Detailed Description	121
7.31.2 Property Documentation	121
7.31.2.1 AlternativeConfigurations	121
7.31.2.2 DefaultConfiguration	121
7.31.2.3 OptionalContentGroups	121
7.32 MuPDFCore.MuPDFOptionalContentGroupLabel Class Reference	122
7.32.1 Detailed Description	122
7.33 MuPDFCore.MuPDFOptionalContentGroupRadioButton Class Reference	122
7.33.1 Detailed Description	123
7.34 MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem Class Reference	123
7.34.1 Detailed Description	124
7.34.2 Member Function Documentation	124
7.34.2.1 Toggle()	124
7.34.3 Property Documentation	124
7.34.3.1 IsEnabled	124
7.34.3.2 IsLocked	125
7.35 MuPDFCore.MuPDFOptionalContentGroupUIItem Class Reference	125
7.35.1 Detailed Description	125
7.35.2 Property Documentation	125
7.35.2.1 Children	126

7.35.2.2 Label	126
7.36 MuPDFCore.MuPDFOutline Class Reference	126
7.36.1 Detailed Description	127
7.36.2 Member Function Documentation	127
7.36.2.1 GetEnumerator()	127
7.36.3 Property Documentation	127
7.36.3.1 Count	127
7.36.3.2 Items	127
7.36.3.3 this[int index]	128
7.37 MuPDFCore.MuPDFOutlineItem Class Reference	128
7.37.1 Detailed Description	128
7.37.2 Property Documentation	128
7.37.2.1 Chapter	129
7.37.2.2 Children	129
7.37.2.3 Location	129
7.37.2.4 Page	129
7.37.2.5 PageNumber	129
7.37.2.6 Title	130
7.37.2.7 Uri	130
7.38 MuPDFCore.MuPDFPage Class Reference	130
7.38.1 Detailed Description	131
7.38.2 Member Function Documentation	131
7.38.2.1 Dispose()	131
7.38.2.2 GetBoundingBox()	131
7.38.3 Property Documentation	132
7.38.3.1 Bounds	132
7.38.3.2 Links	132
7.38.3.3 PageNumber	132
7.39 MuPDFCore.MuPDFPageCollection Class Reference	133
7.39.1 Detailed Description	133
7.39.2 Member Function Documentation	133
7.39.2.1 Dispose()	134
7.39.2.2 GetEnumerator()	134
7.39.3 Property Documentation	134
7.39.3.1 Count	134
7.39.3.2 Length	134
7.39.3.3 this[int index]	134
7.40 MuPDFCore.MuPDFSetOCGStateLinkDestination Class Reference	135
7.40.1 Detailed Description	136
7.40.2 Member Function Documentation	136
7.40.2.1 Activate()	136
7.40.3 Property Documentation	136

7.40.3.1 Type	136
7.41 MuPDFCore.PDFCreationOptions Class Reference	136
7.41.1 Detailed Description	137
7.41.2 Member Enumeration Documentation	137
7.41.2.1 CompressionOptions	138
7.41.2.2 Encryption	138
7.41.2.3 GarbageCollectionOption	138
7.41.3 Property Documentation	138
7.41.3.1 AsciiEncode	138
7.41.3.2 Clean	138
7.41.3.3 CompressStreams	139
7.41.3.4 EncryptionType	139
7.41.3.5 Garbage	139
7.41.3.6 IncludeAnnotations	139
7.41.3.7 Linearize	139
7.41.3.8 OwnerPassword	140
7.41.3.9 Permissions	140
7.41.3.10 PrettyPrint	140
7.41.3.11 RegenerateId	140
7.41.3.12 Sanitize	140
7.41.3.13 UserPassword	141
7.42 MuPDFCore.MuPDFRenderer.PDFRenderer Class Reference	141
7.42.1 Detailed Description	145
7.42.2 Member Enumeration Documentation	145
7.42.2.1 PointerEventHandlers	145
7.42.3 Constructor & Destructor Documentation	145
7.42.3.1 PDFRenderer()	145
7.42.4 Member Function Documentation	145
7.42.4.1 Contain()	146
7.42.4.2 Cover()	146
7.42.4.3 GetProgress()	146
7.42.4.4 GetSelectedText()	146
7.42.4.5 Initialize() [1/4]	146
7.42.4.6 Initialize() [2/4]	147
7.42.4.7 Initialize() [3/4]	148
7.42.4.8 Initialize() [4/4]	148
7.42.4.9 InitializeAsync() [1/4]	149
7.42.4.10 InitializeAsync() [2/4]	150
7.42.4.11 InitializeAsync() [3/4]	151
7.42.4.12 InitializeAsync() [4/4]	151
7.42.4.13 ReleaseResources()	152
7.42.4.14 Render()	152

7.42.4.15 Search()	153
7.42.4.16 SelectAll()	153
7.42.4.17 SetDisplayAreaNow()	153
7.42.4.18 ZoomStep()	153
7.42.5 Member Data Documentation	154
7.42.5.1 ActivateLinksProperty	154
7.42.5.2 BackgroundProperty	154
7.42.5.3 DisplayAreaProperty	154
7.42.5.4 DrawLinksProperty	155
7.42.5.5 HighlightBrushProperty	155
7.42.5.6 HighlightedRegionsProperty	155
7.42.5.7 IsViewerInitializedProperty	155
7.42.5.8 LinkBrushProperty	156
7.42.5.9 LinkPenProperty	156
7.42.5.10 PageBackgroundProperty	156
7.42.5.11 PageNumberProperty	156
7.42.5.12 PageSizeProperty	157
7.42.5.13 PointerEventHandlerTypeProperty	157
7.42.5.14 RenderThreadCountProperty	157
7.42.5.15 SelectionBrushProperty	157
7.42.5.16 SelectionProperty	158
7.42.5.17 ZoomEnabledProperty	158
7.42.5.18 ZoomIncrementProperty	158
7.42.5.19 ZoomProperty	158
7.42.6 Property Documentation	158
7.42.6.1 ActivateLinks	159
7.42.6.2 Background	159
7.42.6.3 DisplayArea	159
7.42.6.4 DrawLinks	159
7.42.6.5 HighlightBrush	159
7.42.6.6 HighlightedRegions	160
7.42.6.7 IsViewerInitialized	160
7.42.6.8 LinkBrush	160
7.42.6.9 LinkPen	160
7.42.6.10 PageBackground	160
7.42.6.11 PageNumber	161
7.42.6.12 PageSize	161
7.42.6.13 PointerEventHandlersType	161
7.42.6.14 RenderThreadCount	161
7.42.6.15 Selection	161
7.42.6.16 SelectionBrush	162
7.42.6.17 Zoom	162

7.42.6.18 ZoomEnabled	162
7.42.6.19 ZoomIncrement	162
7.42.7 Event Documentation	162
7.42.7.1 LinkClicked	163
7.43 MuPDFCore.PointF Struct Reference	163
7.43.1 Detailed Description	163
7.43.2 Constructor & Destructor Documentation	163
7.43.2.1 PointF()	163
7.43.3 Member Data Documentation	164
7.43.3.1 X	164
7.43.3.2 Y	164
7.44 MuPDFCore.Quad Struct Reference	164
7.44.1 Detailed Description	165
7.44.2 Constructor & Destructor Documentation	165
7.44.2.1 Quad()	165
7.44.3 Member Function Documentation	165
7.44.3.1 Contains()	166
7.44.4 Member Data Documentation	166
7.44.4.1 LowerLeft	166
7.44.4.2 LowerRight	166
7.44.4.3 UpperLeft	166
7.44.4.4 UpperRight	167
7.45 MuPDFCore.Rectangle Struct Reference	167
7.45.1 Detailed Description	168
7.45.2 Constructor & Destructor Documentation	168
7.45.2.1 Rectangle() [1/2]	168
7.45.2.2 Rectangle() [2/2]	168
7.45.3 Member Function Documentation	169
7.45.3.1 Contains() [1/2]	169
7.45.3.2 Contains() [2/2]	169
7.45.3.3 Intersect()	170
7.45.3.4 Round() [1/2]	170
7.45.3.5 Round() [2/2]	170
7.45.3.6 Split()	171
7.45.3.7 ToQuad()	171
7.45.4 Member Data Documentation	171
7.45.4.1 X0	172
7.45.4.2 X1	172
7.45.4.3 Y0	172
7.45.4.4 Y1	172
7.45.5 Property Documentation	172
7.45.5.1 Height	172

7.45.5.2 Width	173
7.46 Avalonia.Animation.RectTransition Class Reference	173
7.46.1 Detailed Description	173
7.47 MuPDFCore.RenderProgress Class Reference	173
7.47.1 Detailed Description	174
7.47.2 Property Documentation	174
7.47.2.1 ThreadRenderProgresses	174
7.48 MuPDFCore.RGBMuPDFColorSpace Class Reference	174
7.48.1 Detailed Description	175
7.48.2 Property Documentation	175
7.48.2.1 NumBytes	175
7.48.2.2 Type	175
7.49 MuPDFCore.RoundedRectangle Struct Reference	175
7.49.1 Detailed Description	176
7.49.2 Constructor & Destructor Documentation	176
7.49.2.1 RoundedRectangle()	176
7.49.3 Member Function Documentation	177
7.49.3.1 Split()	177
7.49.4 Member Data Documentation	177
7.49.4.1 X0	177
7.49.4.2 X1	177
7.49.4.3 Y0	178
7.49.4.4 Y1	178
7.49.5 Property Documentation	178
7.49.5.1 Height	178
7.49.5.2 Width	178
7.50 MuPDFCore.RoundedSize Struct Reference	178
7.50.1 Detailed Description	179
7.50.2 Constructor & Destructor Documentation	179
7.50.2.1 RoundedSize()	179
7.50.3 Member Function Documentation	179
7.50.3.1 Split()	180
7.50.4 Member Data Documentation	180
7.50.4.1 Height	180
7.50.4.2 Width	180
7.51 MuPDFCore.SeparationColorSpace Class Reference	181
7.51.1 Detailed Description	181
7.51.2 Property Documentation	181
7.51.2.1 AlternateColorSpace	181
7.51.2.2 ColorantNames	182
7.51.2.3 NumBytes	182
7.51.2.4 Type	182

7.52 MuPDFCore.Size Struct Reference	182
7.52.1 Detailed Description	183
7.52.2 Constructor & Destructor Documentation	183
7.52.2.1 Size() [1/2]	183
7.52.2.2 Size() [2/2]	183
7.52.3 Member Function Documentation	183
7.52.3.1 Split()	184
7.52.4 Member Data Documentation	184
7.52.4.1 Height	184
7.52.4.2 Width	184
7.53 MuPDFCore.SVGCreationOptions Class Reference	184
7.53.1 Detailed Description	185
7.53.2 Member Enumeration Documentation	185
7.53.2.1 TextOption	185
7.53.3 Property Documentation	185
7.53.3.1 IncludeAnnotations	185
7.53.3.2 ReuseImages	186
7.53.3.3 TextRendering	186
7.54 MuPDFCore.TesseractLanguage Class Reference	186
7.54.1 Detailed Description	187
7.54.2 Member Enumeration Documentation	187
7.54.2.1 Best	187
7.54.2.2 BestScripts	188
7.54.2.3 Fast	188
7.54.2.4 FastScripts	188
7.54.3 Constructor & Destructor Documentation	188
7.54.3.1 TesseractLanguage() [1/6]	188
7.54.3.2 TesseractLanguage() [2/6]	189
7.54.3.3 TesseractLanguage() [3/6]	189
7.54.3.4 TesseractLanguage() [4/6]	189
7.54.3.5 TesseractLanguage() [5/6]	190
7.54.3.6 TesseractLanguage() [6/6]	190
7.54.4 Property Documentation	191
7.54.4.1 Language	191
7.54.4.2 Prefix	191
7.55 MuPDFCore.RenderProgress.ThreadRenderProgress Struct Reference	191
7.55.1 Detailed Description	191
7.55.2 Member Data Documentation	192
7.55.2.1 MaxProgress	192
7.55.2.2 Progress	192
7.56 MuPDFCore.TXTCreationOptions Class Reference	192
7.56.1 Detailed Description	193

7.56.2 Property Documentation	193
7.56.2.1 Dehyphenate	193
7.56.2.2 IncludeAnnotations	193
7.56.2.3 IncludeCharactersOutsideMediaBox	193
7.56.2.4 InhibitSpaces	193
7.56.2.5 PreserveLigatures	194
7.56.2.6 PreserveSpans	194
7.56.2.7 PreserveWhitespace	194
7.56.2.8 UseCIDForUnknownUnicode	194
8 File Documentation	195
8.1 PDFRenderer.cs	195
8.2 PDFRenderer.Properties.cs	216
8.3 RectTransition.cs	221
8.4 MuPDF.cs	222
8.5 MuPDFColorSpace.cs	251
8.6 MuPDFContext.cs	256
8.7 MuPDFDisplayList.cs	259
8.8 MuPDFDocument.Create.cs	260
8.9 MuPDFDocument.cs	281
8.10 MuPDFFont.cs	304
8.11 MuPDFImage.cs	306
8.12 MuPDFLinks.cs	313
8.13 MuPDFMultiThreadedPageRenderer.cs	318
8.14 MuPDFOptionalContentGroupConfiguration.cs	326
8.15 MuPDFOutline.cs	332
8.16 MuPDFPage.cs	335
8.17 Rectangles.cs	339
8.18 TesseractLanguage.cs	347
8.19 Utils.cs	366
Index	371

Chapter 1

MuPDFCore: Multiplatform .NET bindings for MuPDF

MuPDFCore is a set of multiplatform .NET bindings for [MuPDF](#). It can render PDF, XPS, EPUB and other formats to raster images returned either as raw bytes, or as image files in multiple formats (including PNG, JPEG, and PSD). It also supports multithreading.

It also includes **MuPDFCore.MuPDFRenderer**, an [Avalonia](#) control to display documents compatible with **MuPDFCore** in [Avalonia windows](#) (with multithreaded rendering).

The library is released under the [AGPLv3](#) licence.

1.1 Getting started

The **MuPDFCore** library targets .NET Standard 2.0, thus it can be used in projects that target .NET Standard 2.0+, .NET Core 2.0+, .NET 5.0+, .NET Framework 4.6.1 (note) and possibly others. **MuPDFCore** includes a pre-compiled native library, which currently supports the following platforms:

- Windows x86 (32 bit) `win-x86`
- Windows x64 (64 bit) `win-x64`
- Windows arm64 (ARM 64 bit) `win-arm64`
- Linux x64 (64 bit)
 - glibc-based `linux-x64`
 - musl-based `linux-musl-x64`
- Linux arm64/aarch64 (ARM 64 bit)
 - glibc-based `linux-arm64`
 - musl-based `linux-musl-arm64` (see note)
- macOS Intel x86_64 (64 bit) `osx-x64`
- macOS Apple silicon (ARM 64 bit) `osx-arm64`

To use the library in your project, you should install the [MuPDFCore NuGet package](#) and/or the [MuPDFCore.PDFRenderer NuGet package](#). When you publish a program that uses [MuPDFCore](#), the correct native library for the target architecture will automatically be copied to the build folder (but see the note for .NET Framework).

Note: you should make sure that end users on [Windows](#) install the [Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017, 2019 and 2022](#) for their platform, otherwise they will get an error message stating that `MuPDFWrapper.dll` could not be loaded because a module was not found.

Note for **musl-based Linux arm64:** I could not find a way to ensure that the `linux-musl-arm64` native artifact overwrites the `linux-arm64` (`glibc`) artifact. As a result, when you publish a project that uses [MuPDFCore](#) targeting `linux-musl-arm64`, you will find *two* native assets in the build directory (`MuPDFWrapper.so`, which is the `musl` artifact, and `libMuPDFWrapper.so`, which is the `glibc` artifact). Everything will work fine out of the box (because the name of the `musl` artifact has higher priority), but you may want to delete `libMuPDFWrapper.so` in order to reduce size. You can use e.g. a post-build target to do this.

1.2 Usage

1.2.1 Documentation

You can find detailed descriptions of how to use [MuPDFCore](#) and some code examples in the [MuPDFCore wiki](#). Interactive documentation for the library API can be accessed from the [documentation website](#). A [PDF reference manual](#) is also available.

1.2.2 Minimal working example

The following example shows the bare minimum code necessary to render a page of a PDF document to a PNG image using [MuPDFCore](#):

```
//Initialise the MuPDF context. This is needed to open or create documents.
using MuPDFContext ctx = new MuPDFContext();
//Open a PDF document
using MuPDFDocument document = new MuPDFDocument(ctx, "path/to/document.pdf");
//Page index (page 0 is the first page of the document)
int pageIndex = 0;
//Zoom level, converting document units into pixels. For a PDF document, a 1x zoom level corresponds to a
//72dpi resolution.
double zoomLevel = 1;
//Save the first page as a PNG image with transparency, at a 1x zoom level (1pt = 1px).
document.SaveImage(pageIndex, zoomLevel, PixelFormats.RGBA, "path/to/output.png",
    RasterOutputFileTypes.PNG);
```

Look at the [wiki](#) for more information.

1.2.3 Examples

The [Demo](#) folder in the repository contains some examples of how the library can be used to extract pages from a PDF or XPS document, render them to a raster image, or combine them in a new document

The [PDFViewerDemo](#) folder contains a complete (though minimal) example of a PDF viewer program built around the [MuPDFCore.MuPDFRenderer.PDFRenderer](#) control.

Note that these examples intentionally avoid any error handling code: in a production setting, you should typically make sure that calls to [MuPDFCore](#) library functions are within a `try...catch` block to handle any resulting [MuPDFExceptions](#).

1.3 Building from source

Building the [MuPDFCore](#) library from source requires the following steps:

1. Building the `libmupdf` native library
2. Building the `MuPDFWrapper` native library
3. Creating the `MuPDFCore.NativeAssets.xxx-yyy` native assets NuGet packages
4. Creating the [MuPDFCore](#) library NuGet package

Starting from [MuPDFCore](#) 1.8.0, the native assets are split into their own NuGet packages, on which the main [MuPDFCore](#) package depends. Aside from reducing the size of individual packages, this means that if you are making changes that do not affect the native assets, you can skip steps 1-3 and go straight to step 4.

Steps 1 and 2 need to be performed on all of Windows, macOS and Linux, and on the various possible architectures (x86, x64 and arm64 for Windows, x64/Intel and arm64/Apple for macOS, and x64 and arm64 for Linux, both glibc and musl - no cross-compiling)! Otherwise, some native assets will be missing and it will not be possible to build the NuGet packages in step 3.

1.3.1 1. Building libmupdf

Starting from version 2.0.0, [MuPDFCore](#) is based on a fork of [MuPDF](#), providing some bugfixes/new features and simplifying the build process.

You can access this fork (currently based on MuPDF 1.25.2) from [arklumpus/mupdf](#). You will need to follow the instructions in that repository to compile the library on Windows, macOS and Linux. You need the following files:

- From Windows (x86, x64, arm64):
 - `libmupdf.lib`
- From macOS (Intel - x64, Apple silicon - arm64):
 - `libmupdf.a`
 - `libmupdf-third.a`
- From Linux (x64, arm64):
 - `libmupdf.a`
 - `libmupdf-third.a`

Note that the files from macOS and Linux are different, despite sharing the same name.

For convenience, these compiled files are included in the `native/MuPDFWrapper/lib` folder of this repository.

1.3.2 2. Building MuPDFWrapper

Once you have the required static library files, you should download the [MuPDFCore](#) source code (just clone this repository) and place the library files in the appropriate subdirectories in the native/MuPDFWrapper/lib/ folder (for Linux x64, copy the library built against glibc to the linux-x64 folder, and the library built against musl to the linux-musl-x64 folder, and do the same for Linux arm64).

To compile MuPDFWrapper you will need [CMake](#) (version 3.8 or higher) and (on Windows) [Ninja](#).

On Windows, the easiest way to get all the required tools is probably to install [Visual Studio](#). By selecting the "Desktop development with C++" workload you should get everything you need.

On macOS, you will need to install at least the Command-Line Tools for Xcode (if necessary, you should be prompted to do this while you perform the following steps) and CMake.

Once you have everything at the ready, you will have to build MuPDFWrapper on the nine platforms.

Build instructions

1.3.2.1 Windows (x86 and x64)

1.

Assuming you have installed Visual Studio, you should open the "x64 Native Tools Command Prompt for VS" or the "x86 Native Tools Command Prompt for VS" (you should be able to find these in the Start menu). Take care to open the version corresponding to the architecture you are building for, otherwise you will not be able to compile the library. A normal command prompt will not work, either.

Note 1: you **must** build the library on two separate systems, one running a 32-bit version of Windows and the other running a 64-bit version. If you try to build the x86 library on an x64 system, the system will probably build a 64-bit library and place it in the 32-bit output folder, which will just make things very confusing.

Note 2 for Windows x86: for some reason, Visual Studio might install the 64-bit version of CMake and Ninja, even though you are on a 32-bit machine. If this happens, you will have to manually install the 32-bit CMake and compile a 32-bit version of Ninja. You will notice if this is an issue because the 64-bit programs will refuse to run.

1. CD to the directory where you have downloaded the [MuPDFCore](#) source code.
2. CD into the native directory.
3. Type build. This will start the build.cmd batch script that will delete any previous build and compile the library.

After this finishes, you should find a file named MuPDFWrapper.dll in the native/out/build/win-x64/← MuPDFWrapper/ directory or in the native/out/build/win-x64/MuPDFWrapper/ directory. Leave it there.

1.3.2.2 Windows (arm64)

1. Locate the batch file that sets up the developer command prompt environment. You can do this by finding the "Developer Command Prompt for VS" link in the start menu, then clicking on Open file location, opening the properties of the link and looking at the Target. This could be e.g. C:\Program Files\Microsoft Visual Studio\2022\Preview\Common7\Tools\VsDevCmd.bat.
2. Open a normal command prompt and invoke the batch script with the `-arch=arm64 -host_arch=x86` arguments (add quotes if there are spaces in the path to the batch script), e.g.:
`"C:\Program Files\Microsoft Visual Studio\2022\Preview\Common7\Tools\VsDevCmd.bat" -arch=arm64 -host_arch=x86`
3. CD to the directory where you have downloaded the [MuPDFCore](#) source code.
4. CD into the native directory.
5. Type build. This will start the `build.cmd` batch script that will delete any previous build and compile the library.

After this finishes, you should find a file named `MuPDFWrapper.dll` in the `native/out/build/win-arm64/` `MuPDFWrapper/` directory. Leave it there.

1.3.2.3 macOS and Linux

1. Assuming you have everything ready, open a terminal in the folder where you have downloaded the [MuPDFCore](#) source code.
2. `cd` into the native directory.
3. Type `chmod +x build.sh`.
4. Type `./build.sh`. This will delete any previous build and compile the library.

After this finishes, you should find a file named `libMuPDFWrapper.dylib` in the `native/out/build/mac-x64/` `MuPDFWrapper/` directory (on macOS running on an Intel x64 processor) or in the `native/out/build/mac-arm64/` `MuPDFWrapper/` directory (on macOS running on an Apple silicon arm64 processor), and a file named `libMuPDFWrapper.so` in the `native/out/build/linux-XXX/MuPDFWrapper/` directory (on Linux - where XXX can be x64, arm64, musl-x64, or musl-arm64). Leave it there.

1.3.3 3. Creating the native assets MuPDFCore NuGet packages

Once you have the `MuPDFWrapper.dll` (3x), `libMuPDFWrapper.dylib` (2x) and `libMuPDFWrapper.so` (4x) files, make sure they are in the correct folders (`native/out/build/xxx-yyyy/MuPDFWrapper/`), **all on the same machine**.

To create the native assets NuGet packages, you will need the [.NET Core 2.0 SDK or higher](#) for your platform. Once you have installed it and have everything ready, open a terminal in the folder where you have downloaded the [MuPDFCore](#) source code and type:

`BuildNativeAssets`

This will create the NuGet packages in the `MuPDFCore.NativeAssets/NuGetPackages` folder. Once the script finishes, this folder should contain 9 files. Make sure you add this folder as a local NuGet source.

1.3.4 4. Creating the MuPDFCore NuGet package

If you have made updates to the native assets, make sure to use the appropriate version numbers in `MuPDFCore/MuPDFCore.csproj`. Then, to create the main `MuPDFCore` NuGet package, open a terminal in the folder where you have downloaded the `MuPDFCore` source code and type:

```
cd MuPDFCore
dotnet pack -c Release
```

This will create a NuGet package in `MuPDFCore/bin/Release`. You can install this package on your projects by adding a local NuGet source.

1.3.5 5. Running tests

To verify that everything is working correctly, you should build the `MuPDFCore` test suite and run it on all platforms. To build the test suite, you will need the `.NET 7 SDK or higher`. You will also need to have enabled the `Windows Subsystem for Linux (WSL)`.

To build the test suite:

1. Make sure that you have changed the version of the `MuPDFCore` NuGet package so that it is higher than the latest version of `MuPDFCore` in the NuGet repository (you should use a pre-release suffix, e.g. `1.4.0-a1` to avoid future headaches with new versions of `MuPDFCore`). This is set in line 9 of the `MuPDFCore/MuPDFCore.csproj` file.
2. Add the `MuPDFCore/bin/Release` folder to your local NuGet repositories (you can do this e.g. in Visual Studio).
3. If you have not done so already, create the `MuPDFCore` NuGet package following step 4 above.
4. Update line 76 of the `Tests/Tests.csproj` project file so that it refers to the version of the `MuPDFCore` package you have just created.

These steps ensure that you are testing the right version of `MuPDFCore` (i.e. your freshly built copy) and not something else that may have been cached.

Now, open a Windows command line in the folder where you have downloaded the `MuPDFCore` source code, type `BuildTests` and press `Enter`. This will create a number of files in the `Release\MuPDFCoreTests` folder, where each file is an archive containing the tests for a certain platform and architecture:

- `MuPDFCoreTests-linux-x64.tar.gz` contains the tests for Linux environments using `glibc` on `x64` processors.
- `MuPDFCoreTests-linux-arm64.tar.gz` contains the tests for Linux environments using `glibc` on `arm64` processors.
- `MuPDFCoreTests-linux-musl-x64.tar.gz` contains the tests for Linux environments using `musl` on `x64` processors.
- `MuPDFCoreTests-linux-musl-arm64.tar.gz` contains the tests for Linux environments using `musl` on `arm64` processors.
- `MuPDFCoreTests-mac-x64.tar.gz` contains the tests for macOS environments on Intel processors.
- `MuPDFCoreTests-mac-arm64.tar.gz` contains the tests for macOS environments on Apple silicon processors.
- `MuPDFCoreTests-win-x64.tar.gz` contains the tests for Windows environments on `x64` processors.
- `MuPDFCoreTests-win-x86.tar.gz` contains the tests for Windows environments on `x86` processors.
- `MuPDFCoreTests-win-arm64.tar.gz` contains the tests for Windows environments on `arm64` processors.

To run the tests, copy each archive to a machine running the corresponding operating system, and extract it (note: on Windows, the default zip file manager may struggle when extracting the text file with non-latin characters; you may need to manually extract this file, use `7-Zip`, or use the `unzip` command from within the WSL). Then:

1.3.5.1 Windows

- Open a command prompt and CD into the folder where you have extracted the contents of the test archive.
- Enter the command `MuPDFCoreTestHost` (this will run the test program).

1.3.5.2 macOS and Linux

- Open a terminal and `cd` into the folder where you have extracted the contents of the test archive.
- Enter the command `chmod +x MuPDFCoreTestHost` (this will add the executable flag to the test program).
- Enter the command `./MuPDFCoreTestHost` (this will run the test program).
- On macOS, depending on your security settings, you may get a message saying `zsh: killed` when you try to run the program. To address this, you need to sign the executable, e.g. by running `codesign --timestamp --sign <certificate> MuPDFCoreTestHost`, where `<certificate>` is the name of a code signing certificate in your keychain (e.g. Developer ID Application: John Smith). After this, you can try again to run the test program with `./MuPDFCoreTestHost`.

The test suite will start; it will print the name of each test, followed by a green `Succeeded` or a red `Failed` depending on the test result. If everything went correctly, all tests should succeed.

When all the tests have been run, the program will print a summary showing how many tests have succeeded (if any) and how many have failed (if any). If any tests have failed, a list of these will be printed, and then they will be run again one at a time, waiting for a key press before running each test (this makes it easier to follow what is going on). If you wish to kill the test process early, you can do so with `CTRL+C`.

1.4 Note about MuPDFCore and .NET Framework

If you wish to use `MuPDFCore` in a .NET Framework project, you will need to manually copy the native `MuPDFWrapper` library for the platform you are using to the executable directory (this is done automatically if you target .NET/.NET core).

One way to obtain the appropriate library files is:

1. Manually download the appropriate native assets NuGet package from the table below. Note that AnyCPU builds on Windows need the `win-x86` native asset.
2. Rename the `.nupkg` file so that it has a `.zip` extension.
3. Extract the zip file.
4. Within the extracted folder, the library files are in the `runtimes/xxx/native/` folder, where `xxx` is `linux-x64`, `linux-arm64`, `linux-musl-x64`, `linux-musl-arm64`, `osx-x64`, `osx-arm64`, `win-x64`, `win-x86` or `win-arm64`, depending on the platform you are using.
5. The file you need to copy should be called `MuPDFWrapper.dll` on Windows, `libMuPDFWrapper.so` or `MuPDFWrapper.so` on Linux, and `libMuPDFWrapper.dylib` on macOS.

Make sure you copy the appropriate file to the same folder as the executable!

OS**Platform****NuGet package** </thead> <tbody>

Windows

x86

win-x86

x64

win-x64

arm64

win-arm64

Linux

x64

glibc

linux-x64

musl

linux-musl-x64

arm64

glibc

linux-arm64

musl

linux-musl-arm64

macOS

x64 (Intel)

osx-x64

arm64 (Apple Silicon)

osx-arm64 </tbody>

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Avalonia	19
Avalonia.Animation	19
MuPDFCore	19
MuPDFCore.MuPDFRenderer	24

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

MuPDFCore.CBZCreationOptions	26
Control	
MuPDFCore.MuPDFRenderer.PDFRenderer	141
MuPDFCore.MuPDFRenderer.PDFRenderer	141
MuPDFCore.MuPDFDocument.Create	31
MuPDFCore.PDFCreationOptions.DocumentPermissions	48
EventArgs	
MuPDFCore.MessageEventArgs	59
MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs	109
Exception	
MuPDFCore.DocumentLockedException	47
MuPDFCore.LifetimeManagementException< T1, T2 >	57
MuPDFCore.MuPDFException	92
MuPDFCore.HTMLCreationOptions	52
IDisposable	
MuPDFCore.DisposableIntPtr	46
MuPDFCore.MuPDFContext	65
MuPDFCore.MuPDFDocument	68
MuPDFCore.MuPDFFont	95
MuPDFCore.MuPDFImage	98
MuPDFCore.MuPDFLinks	112
MuPDFCore.MuPDFMultiThreadedPageRenderer	113
MuPDFCore.MuPDFPage	130
MuPDFCore.MuPDFPageCollection	133
InterpolatingTransitionBase	
Avalonia.Animation.RectTransition	173
IReadOnlyList	
MuPDFCore.MuPDFLinks	112
MuPDFCore.MuPDFOutline	126
MuPDFCore.MuPDFPageCollection	133
MuPDFCore.MuPDF	61
MuPDFCore.MuPDFColorSpace	63
MuPDFCore.BGRMuPDFColorSpace	25
MuPDFCore.CMYKMuPDFColorSpace	30
MuPDFCore.GrayscaleMuPDFColorSpace	51

MuPDFCore.IndexedMuPDFColorSpace	55
MuPDFCore.LabMuPDFColorSpace	56
MuPDFCore.RGBMuPDFColorSpace	174
MuPDFCore.SeparationColorSpace	181
MuPDFCore.MuPDFLink	107
MuPDFCore.MuPDFLinkDestination	110
MuPDFCore.MuPDFExternalLinkDestination	93
MuPDFCore.MuPDFInternalLinkDestination	104
MuPDFCore.MuPDFSetOCGStateLinkDestination	135
MuPDFCore.MuPDFOptionalContentGroup	117
MuPDFCore.MuPDFOptionalContentGroupConfiguration	119
MuPDFCore.MuPDFOptionalContentGroupData	120
MuPDFCore.MuPDFOptionalContentGroupUIItem	125
MuPDFCore.MuPDFOptionalContentGroupLabel	122
MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem	123
MuPDFCore.MuPDFOptionalContentGroupCheckbox	118
MuPDFCore.MuPDFOptionalContentGroupRadioButton	122
MuPDFCore.MuPDFOutlineItem	128
MuPDFCore.PDFCreationOptions	136
MuPDFCore.PointF	163
MuPDFCore.Quad	164
MuPDFCore.Rectangle	167
MuPDFCore.RenderProgress	173
MuPDFCore.RoundedRectangle	175
MuPDFCore.RoundedSize	178
MuPDFCore.Size	182
MuPDFCore.SVGCreationOptions	184
MuPDFCore.TesseractLanguage	186
MuPDFCore.RenderProgress.ThreadRenderProgress	191
MuPDFCore.TXTCreationOptions	192

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

MuPDFCore.BGRMuPDFColorSpace	
Represents a colour space where each pixel is stored as three bytes encoding the B, G, and R components.	25
MuPDFCore.CBZCreationOptions	
Options for creating a CBZ document.	26
MuPDFCore.CMYKMuPDFColorSpace	
Represents a colour space where each pixel is stored as four bytes encoding the C, M, Y, and K components.	30
MuPDFCore.MuPDFDocument.Create	
Contains methods to create documents by combining pages from other documents.	31
MuPDFCore.DisposableIntPtr	
An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.	46
MuPDFCore.DocumentLockedException	
The exception that is thrown when an attempt is made to render an encrypted document without supplying the required password.	47
MuPDFCore.PDFCreationOptions.DocumentPermissions	
Specifies which actions can be performed without the owner password.	48
MuPDFCore.GrayscaleMuPDFColorSpace	
Represents a colour space where each pixel is stored as a single byte.	51
MuPDFCore.HTMLCreationOptions	
Options for creating an HTML or XHTML document.	52
MuPDFCore.IndexedMuPDFColorSpace	
Represents a colour space where each pixel is stored as a single byte mapping the pixel to a colour in the lookup table.	55
MuPDFCore.LabMuPDFColorSpace	
Represents a colour space where each pixel is stored as three bytes encoding the L*, a, and b components.	56
MuPDFCore.LifetimeManagementException< T1, T2 >	
The exception that is thrown when the lifetime of disposable objects is not managed properly.	57
MuPDFCore.MessageEventArgs	
EventArgs for the MuPDF.StandardOutputMessage and MuPDF.StandardErrorMessage events.	59
MuPDFCore.MuPDF	
Contains static methods to perform setup operations.	61
MuPDFCore.MuPDFColorSpace	
Represents a colour space.	63

MuPDFCore.MuPDFContext	A wrapper around a MuPDF context object, which contains the exception stack and the resource cache store.	65
MuPDFCore.MuPDFDocument	A wrapper over a MuPDF document object, which contains possibly multiple pages.	68
MuPDFCore.MuPDFException	The exception that is thrown when a MuPDF operation fails.	92
MuPDFCore.MuPDFExternalLinkDestination	A link destination to an external resource (e.g., a website).	93
MuPDFCore.MuPDFFont	Represents a font.	95
MuPDFCore.MuPDFImage	Represents an image embedded within a document.	98
MuPDFCore.MuPDFInternalLinkDestination	An internal link destination.	104
MuPDFCore.MuPDFLink	Represents a link within a MuPDF document.	107
MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs	EventArgs for the PDFRenderer.LinkClicked event.	109
MuPDFCore.MuPDFLinkDestination	Represents a generic link destination.	110
MuPDFCore.MuPDFLinks	A collection of MuPDFLink objects.	112
MuPDFCore.MuPDFMultiThreadedPageRenderer	A class that holds the necessary resources to render a page of a MuPDF document using multiple threads.	113
MuPDFCore.MuPDFOptionalContentGroup	Represents an optional content group (also known as layer).	117
MuPDFCore.MuPDFOptionalContentGroupCheckbox	An optional content group UI element that should be represented as a check box.	118
MuPDFCore.MuPDFOptionalContentGroupConfiguration	Represents an optional content group configuration.	119
MuPDFCore.MuPDFOptionalContentGroupData	Contains information about optional content groups in a PDF document (also known as layers).	120
MuPDFCore.MuPDFOptionalContentGroupLabel	An optional content group UI element that should be represented as a label.	122
MuPDFCore.MuPDFOptionalContentGroupRadioButton	An optional content group UI element that should be represented as a radio button.	122
MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem	An optional content group UI element that can be enabled or disabled.	123
MuPDFCore.MuPDFOptionalContentGroupUIItem	An optional content group UI element.	125
MuPDFCore.MuPDFOutline	Represents a document outline (table of contents).	126
MuPDFCore.MuPDFOutlineItem	Represents an item in a document outline (table of contents).	128
MuPDFCore.MuPDFPage	A wrapper over a MuPDF page object, which contains information about the page's boundaries.	130
MuPDFCore.MuPDFPageCollection	A lazy collection of MuPDFPages. Each page is loaded from the document as it is requested for the first time.	133
MuPDFCore.MuPDFSetOCGStateLinkDestination	A destination for a link whose effect is to change the visibility state of some optional content groups (also known as layers).	135
MuPDFCore.PDFCreationOptions	Options for creating a PDF document.	136
MuPDFCore.MuPDFRenderer.PDFRenderer	A control to render PDF documents (and other formats), potentially using multiple threads.	141

MuPDFCore.PointF	Represents a point.	163
MuPDFCore.Quad	Represents a quadrilateral (not necessarily a rectangle).	164
MuPDFCore.Rectangle	Represents a rectangle.	167
Avalonia.Animation.RectTransition	Transition class that handles AvaloniaProperty with Rect types.	173
MuPDFCore.RenderProgress	Holds a summary of the progress of the current rendering operation.	173
MuPDFCore.RGBMuPDFColorSpace	Represents a colour space where each pixel is stored as three bytes encoding the R, G, and B components.	174
MuPDFCore.RoundedRectangle	Represents a rectangle using only integer numbers.	175
MuPDFCore.RoundedSize	Represents the size of a rectangle using only integer numbers.	178
MuPDFCore.SeparationColorSpace	Represents a separation colour space.	181
MuPDFCore.Size	Represents the size of a rectangle.	182
MuPDFCore.SVGCreationOptions	Options for creating an SVG document.	184
MuPDFCore.TesseractLanguage	Represents a language used by Tesseract OCR.	186
MuPDFCore.RenderProgress.ThreadRenderProgress	Holds the progress of a single thread.	191
MuPDFCore.TXTCreationOptions	Options for creating a text or structured text document.	192

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

MuPDFCore.MuPDFRenderer/PDFRenderer.cs	195
MuPDFCore.MuPDFRenderer/PDFRenderer.Properties.cs	216
MuPDFCore.MuPDFRenderer/RectTransition.cs	221
MuPDFCore/MuPDF.cs	222
MuPDFCore/MuPDFColorSpace.cs	251
MuPDFCore/MuPDFContext.cs	256
MuPDFCore/MuPDFDisplayList.cs	259
MuPDFCore/MuPDFDocument.Create.cs	260
MuPDFCore/MuPDFDocument.cs	281
MuPDFCore/MuPDFFont.cs	304
MuPDFCore/MuPDFImage.cs	306
MuPDFCore/MuPDFLinks.cs	313
MuPDFCore/MuPDFMultiThreadedPageRenderer.cs	318
MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs	326
MuPDFCore/MuPDFOutline.cs	332
MuPDFCore/MuPDFPage.cs	335
MuPDFCore/Rectangles.cs	339
MuPDFCore/TesseractLanguage.cs	347
MuPDFCore/Utils.cs	366

Chapter 6

Namespace Documentation

6.1 Avalonia Namespace Reference

6.2 Avalonia.Animation Namespace Reference

Classes

- class [RectTransition](#)
Transition class that handles `AvaloniaProperty` with `Rect` types.

6.3 MuPDFCore Namespace Reference

Classes

- class [BGRMuPDFColorSpace](#)
Represents a colour space where each pixel is stored as three bytes encoding the `B`, `G`, and `R` components.
- class [CBZCreationOptions](#)
Options for creating a CBZ document.
- class [CMYKMuPDFColorSpace](#)
Represents a colour space where each pixel is stored as four bytes encoding the `C`, `M`, `Y`, and `K` components.
- class [DisposableIntPtr](#)
An `IDisposable` wrapper around an `IntPtr` that frees the allocated memory when it is disposed.
- class [DocumentLockedException](#)
The exception that is thrown when an attempt is made to render an encrypted document without supplying the required password.
- class [GrayscaleMuPDFColorSpace](#)
Represents a colour space where each pixel is stored as a single byte.
- class [HTMLCreationOptions](#)
Options for creating an HTML or XHTML document.
- class [IndexedMuPDFColorSpace](#)
Represents a colour space where each pixel is stored as a single byte mapping the pixel to a colour in the lookup table.
- class [LabMuPDFColorSpace](#)

- class [LifetimeManagementException](#)

Represents a colour space where each pixel is stored as three bytes encoding the L, a, and b components.*
- class [MessageEventArgs](#)

The exception that is thrown when the lifetime of disposable objects is not managed properly.
- class [MuPDF](#)

EventArgs for the MuPDF.StandardOutputMessage and MuPDF.StandardErrorMessage events.
- class [MuPDFColorSpace](#)

Contains static methods to perform setup operations.
- class [MuPDFContext](#)

Represents a colour space.
- class [MuPDFDocument](#)

A wrapper around a MuPDF context object, which contains the exception stack and the resource cache store.
- class [MuPDFException](#)

The exception that is thrown when a MuPDF operation fails.
- class [MuPDFExternalLinkDestination](#)

A link destination to an external resource (e.g., a website).
- class [MuPDFFont](#)

Represents a font.
- class [MuPDFImage](#)

Represents an image embedded within a document.
- class [MuPDFInternalLinkDestination](#)

An internal link destination.
- class [MuPDFLink](#)

Represents a link within a MuPDF document.
- class [MuPDFLinkDestination](#)

Represents a generic link destination.
- class [MuPDFLinks](#)

A collection of MuPDFLink objects.
- class [MuPDFMultiThreadedPageRenderer](#)

A class that holds the necessary resources to render a page of a MuPDF document using multiple threads.
- class [MuPDFOptionalContentGroup](#)

Represents an optional content group (also known as layer).
- class [MuPDFOptionalContentGroupCheckbox](#)

An optional content group UI element that should be represented as a check box.
- class [MuPDFOptionalContentGroupConfiguration](#)

Represents an optional content group configuration.
- class [MuPDFOptionalContentGroupData](#)

Contains information about optional content groups in a PDF document (also known as layers).
- class [MuPDFOptionalContentGroupLabel](#)

An optional content group UI element that should be represented as a label.
- class [MuPDFOptionalContentGroupRadioButton](#)

An optional content group UI element that should be represented as a radio button.
- class [MuPDFOptionalContentGroupSelectableUIItem](#)

An optional content group UI element that can be enabled or disabled.
- class [MuPDFOutline](#)

Represents a document outline (table of contents).
- class [MuPDFOutlineItem](#)

Represents an item in a document outline (table of contents).

- class [MuPDFPage](#)
A wrapper over a MuPDF page object, which contains information about the page's boundaries.
- class [MuPDFPageCollection](#)
A lazy collection of MuPDFPages. Each page is loaded from the document as it is requested for the first time.
- class [MuPDFSetOCGStateLinkDestination](#)
A destination for a link whose effect is to change the visibility state of some optional content groups (also known as layers).
- class [PDFCreationOptions](#)
Options for creating a PDF document.
- struct [PointF](#)
Represents a point.
- struct [Quad](#)
Represents a quadrilateral (not necessarily a rectangle).
- struct [Rectangle](#)
Represents a rectangle.
- class [RenderProgress](#)
Holds a summary of the progress of the current rendering operation.
- class [RGBMuPDFColorSpace](#)
Represents a colour space where each pixel is stored as three bytes encoding the R, G, and B components.
- struct [RoundedRectangle](#)
Represents a rectangle using only integer numbers.
- struct [RoundedSize](#)
Represents the size of a rectangle using only integer numbers.
- class [SeparationColorSpace](#)
Represents a separation colour space.
- struct [Size](#)
Represents the size of a rectangle.
- class [SVGCreationOptions](#)
Options for creating an SVG document.
- class [TesseractLanguage](#)
Represents a language used by Tesseract OCR.
- class [TXTCreationOptions](#)
Options for creating a text or structured text document.

Enumerations

- enum [ExitCodes](#)
Exit codes returned by native methods describing various errors that can occur.
- enum [InputFileTypes](#)
File types supported in input by the library.
- enum [RasterOutputFileTypes](#)
Raster image file types supported in output by the library.
- enum [DocumentOutputFileTypes](#)
Document file types supported in output by the library.
- enum [PixelFormats](#)
Pixel formats supported by the library.
- enum [EncryptionState](#)
Possible document encryption states.
- enum [RestrictionState](#)
Possible document restriction states.

- enum [DocumentRestrictions](#)
Document restrictions.
- enum [PasswordTypes](#)
Password types.
- enum [BoxType](#)
Types of bounding boxes.
- enum [ColorSpaceType](#)
Colour space used by the image.

6.3.1 Enumeration Type Documentation

6.3.1.1 BoxType

enum [MuPDFCore.BoxType](#)

Types of bounding boxes.

Definition at line [436](#) of file [MuPDF.cs](#).

6.3.1.2 ColorSpaceType

enum [MuPDFCore.ColorSpaceType](#)

Colour space used by the image.

Definition at line [27](#) of file [MuPDFColorSpace.cs](#).

6.3.1.3 DocumentOutputFileTypes

enum [MuPDFCore.DocumentOutputFileTypes](#)

Document file types supported in output by the library.

Definition at line [275](#) of file [MuPDF.cs](#).

6.3.1.4 DocumentRestrictions

enum [MuPDFCore.DocumentRestrictions](#)

Document restrictions.

Definition at line [384](#) of file [MuPDF.cs](#).

6.3.1.5 EncryptionState

enum [MuPDFCore.EncryptionState](#)

Possible document encryption states.

Definition at line [342](#) of file [MuPDF.cs](#).

6.3.1.6 ExitCodes

enum [MuPDFCore.ExitCodes](#)

Exit codes returned by native methods describing various errors that can occur.

Definition at line [32](#) of file [MuPDF.cs](#).

6.3.1.7 InputFileTypes

enum [MuPDFCore.InputFileTypes](#)

File types supported in input by the library.

Definition at line [148](#) of file [MuPDF.cs](#).

6.3.1.8 PasswordTypes

enum [MuPDFCore.PasswordTypes](#)

Password types.

Definition at line [415](#) of file [MuPDF.cs](#).

6.3.1.9 PixelFormats

enum [MuPDFCore.PixelFormats](#)

Pixel formats supported by the library.

Definition at line [316](#) of file [MuPDF.cs](#).

6.3.1.10 RasterOutputFileTypes

enum [MuPDFCore.RasterOutputFileTypes](#)

Raster image file types supported in output by the library.

Definition at line [244](#) of file [MuPDF.cs](#).

6.3.1.11 RestrictionState

enum [MuPDFCore.RestrictionState](#)

Possible document restriction states.

Definition at line [363](#) of file [MuPDF.cs](#).

6.4 MuPDFCore.MuPDFRenderer Namespace Reference

Classes

- class [MupdfLinkClickedEventArgs](#)
EventArgs for the [PDFRenderer.LinkClicked](#) event.
- class [PDFRenderer](#)
A control to render PDF documents (and other formats), potentially using multiple threads.

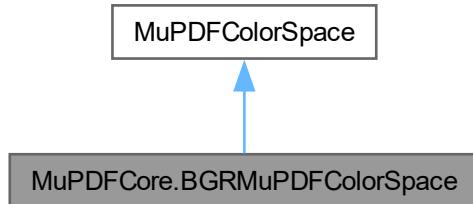
Chapter 7

Class Documentation

7.1 MuPDFCore.BGRMuPDFColorSpace Class Reference

Represents a colour space where each pixel is stored as three bytes encoding the B, G, and R components.

Inheritance diagram for MuPDFCore.BGRMuPDFColorSpace:



Properties

- override `ColorSpaceType Type` [get]
- override int `NumBytes` [get]

Additional Inherited Members

7.1.1 Detailed Description

Represents a colour space where each pixel is stored as three bytes encoding the B, G, and R components.

Definition at line 273 of file [MuPDFColorSpace.cs](#).

7.1.2 Property Documentation

7.1.2.1 NumBytes

`override int MuPDFCore.BGRMuPDFColorSpace.NumBytes [get]`

Definition at line 279 of file [MuPDFColorSpace.cs](#).

7.1.2.2 Type

`override ColorSpaceType MuPDFCore.BGRMuPDFColorSpace.Type [get]`

Definition at line 276 of file [MuPDFColorSpace.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFColorSpace.cs](#)

7.2 MuPDFCore.CBZCreationOptions Class Reference

Options for creating a CBZ document.

Public Types

- enum [ColorSpace](#)
Colour spaces for the rendered images.
- enum [RasterizerOption](#)
Rasterizer settings.

Properties

- bool [IncludeAnnotations](#) = true [get, set]
If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.
- double [Rotate](#) = double.NaN [get, set]
Angle (in degrees) by which the rendered pages will be rotated.
- double [XResolution](#) = double.NaN [get, set]
X resolution of the rendered pages in pixels per inch.
- double [YResolution](#) = double.NaN [get, set]
Y resolution of the rendered pages in pixels per inch.
- double [Width](#) = double.NaN [get, set]
Render pages to fit the specified width.
- double [Height](#) = double.NaN [get, set]
Render pages to fit the specified height.
- [ColorSpace](#) [RenderingColorSpace](#) = [ColorSpace.RGB](#) [get, set]
Colour space for rendering.
- bool [Alpha](#) = false [get, set]
Render pages with an alpha channel and transparent background.
- [RasterizerOption](#) [GraphicsRasterizer](#) = [RasterizerOption.Default](#) [get, set]
Rasterizer settings for graphics elements.
- [RasterizerOption](#) [TextRasterizer](#) = [RasterizerOption.Default](#) [get, set]
Rasterizer settings for text elements.

7.2.1 Detailed Description

Options for creating a CBZ document.

Definition at line 613 of file [MuPDFDocument.Create.cs](#).

7.2.2 Member Enumeration Documentation

7.2.2.1 ColorSpace

```
enum MuPDFCore.CBZCreationOptions.ColorSpace
```

Colour spaces for the rendered images.

Definition at line 623 of file [MuPDFDocument.Create.cs](#).

7.2.2.2 RasterizerOption

```
enum MuPDFCore.CBZCreationOptions.RasterizerOption
```

Rasterizer settings.

Definition at line 644 of file [MuPDFDocument.Create.cs](#).

7.2.3 Property Documentation

7.2.3.1 Alpha

```
bool MuPDFCore.CBZCreationOptions.Alpha = false [get], [set]
```

Render pages with an alpha channel and transparent background.

Definition at line 741 of file [MuPDFDocument.Create.cs](#).

7.2.3.2 GraphicsRasterizer

```
RasterizerOption MuPDFCore.CBZCreationOptions.GraphicsRasterizer = RasterizerOption.Default  
[get], [set]
```

Rasterizer settings for graphics elements.

Definition at line 746 of file [MuPDFDocument.Create.cs](#).

7.2.3.3 Height

```
double MuPDFCore.CBZCreationOptions.Height = double.NaN [get], [set]
```

Render pages to fit the specified height.

Definition at line 731 of file [MuPDFDocument.Create.cs](#).

7.2.3.4 IncludeAnnotations

```
bool MuPDFCore.CBZCreationOptions.IncludeAnnotations = true [get], [set]
```

If this is `true`, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.

Definition at line 618 of file [MuPDFDocument.Create.cs](#).

7.2.3.5 RenderingColorSpace

```
ColorSpace MuPDFCore.CBZCreationOptions.RenderingColorSpace = ColorSpace.RGB [get], [set]
```

Colour space for rendering.

Definition at line 736 of file [MuPDFDocument.Create.cs](#).

7.2.3.6 Rotate

```
double MuPDFCore.CBZCreationOptions.Rotate = double.NaN [get], [set]
```

Angle (in degrees) by which the rendered pages will be rotated.

Definition at line 711 of file [MuPDFDocument.Create.cs](#).

7.2.3.7 TextRasterizer

```
RasterizerOption MuPDFCore.CBZCreationOptions.TextRasterizer = RasterizerOption.Default [get],  
[set]
```

Rasterizer settings for text elements.

Definition at line 751 of file [MuPDFDocument.Create.cs](#).

7.2.3.8 Width

```
double MuPDFCore.CBZCreationOptions.Width = double.NaN [get], [set]
```

Render pages to fit the specified width.

Definition at line 726 of file [MuPDFDocument.Create.cs](#).

7.2.3.9 XResolution

```
double MuPDFCore.CBZCreationOptions.XResolution = double.NaN [get], [set]
```

X resolution of the rendered pages in pixels per inch.

Definition at line 716 of file [MuPDFDocument.Create.cs](#).

7.2.3.10 YResolution

```
double MuPDFCore.CBZCreationOptions.YResolution = double.NaN [get], [set]
```

Y resolution of the rendered pages in pixels per inch.

Definition at line 721 of file [MuPDFDocument.Create.cs](#).

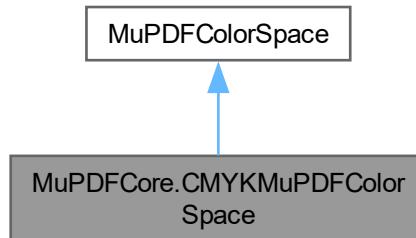
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFDocument.Create.cs

7.3 MuPDFCore.CMYKMuPDFColorSpace Class Reference

Represents a colour space where each pixel is stored as four bytes encoding the C, M, Y, and K components.

Inheritance diagram for MuPDFCore.CMYKMuPDFColorSpace:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]

Additional Inherited Members

7.3.1 Detailed Description

Represents a colour space where each pixel is stored as four bytes encoding the C, M, Y, and K components.

Definition at line [307](#) of file [MuPDFColorSpace.cs](#).

7.3.2 Property Documentation

7.3.2.1 NumBytes

```
override int MuPDFCore.CMYKMuPDFColorSpace.NumBytes [get]
```

Definition at line [313](#) of file [MuPDFColorSpace.cs](#).

7.3.2.2 Type

```
override ColorSpaceType MuPDFCore.CMYKMuPDFColorSpace.Type [get]
```

Definition at line 310 of file [MuPDFColorSpace.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFColorSpace.cs](#)

7.4 MuPDFCore.MuPDFDocument.Create Class Reference

Contains methods to create documents by combining pages from other documents.

Static Public Member Functions

- static void [Document](#) ([MuPDFContext](#) context, string fileName, [DocumentOutputFileTypes](#) fileType, [IEnumerable<\(MuPDFPage page, Rectangle region, float zoom\)>](#) pages, bool includeAnnotations=true)
Create a new document containing the specified (parts of) pages from other documents.
- static void [Document](#) ([MuPDFContext](#) context, string fileName, [DocumentOutputFileTypes](#) fileType, params([MuPDFPage](#) page, [Rectangle](#) region, float zoom)[] pages)
Create a new document containing the specified (parts of) pages from other documents.
- static void [Document](#) ([MuPDFContext](#) context, string fileName, [DocumentOutputFileTypes](#) fileType, [IEnumerable< MuPDFPage >](#) pages, bool includeAnnotations=true)
Create a new document containing the specified (parts of) pages from other documents.
- static void [Document](#) ([MuPDFContext](#) context, string fileName, [DocumentOutputFileTypes](#) fileType, params [MuPDFPage\[\]](#) pages)
Create a new document containing the specified (parts of) pages from other documents.
- static void [PDFDocument](#) ([MuPDFContext](#) context, string fileName, [IEnumerable<\(MuPDFPage page, Rectangle region, float zoom\)>](#) pages, [PDFCreationOptions](#) options=default)
Create a new PDF document containing the specified (parts of) pages from other documents.
- static void [PDFDocument](#) ([MuPDFContext](#) context, string fileName, params([MuPDFPage](#) page, [Rectangle](#) region, float zoom)[] pages)
Create a new PDF document containing the specified (parts of) pages from other documents.
- static void [PDFDocument](#) ([MuPDFContext](#) context, string fileName, [IEnumerable< MuPDFPage >](#) pages, [PDFCreationOptions](#) options=default)
Create a new PDF document containing the specified (parts of) pages from other documents.
- static void [PDFDocument](#) ([MuPDFContext](#) context, string fileName, params [MuPDFPage\[\]](#) pages)
Create a new PDF document containing the specified (parts of) pages from other documents.
- static void [SVGDocument](#) ([MuPDFContext](#) context, string fileName, [MuPDFPage](#) page, [Rectangle](#) region, float zoom, [SVGCreationOptions](#) options=default)
Create a new SVG document containing the specified (parts of) pages from other documents.
- static void [SVGDocument](#) ([MuPDFContext](#) context, string fileName, [MuPDFPage](#) page, [SVGCreationOptions](#) options=default)
Create a new SVG document containing the specified (parts of) pages from other documents.
- static void [CBZDocument](#) ([MuPDFContext](#) context, string fileName, [IEnumerable<\(MuPDFPage page, Rectangle region, float zoom\)>](#) pages, [CBZCreationOptions](#) options=default)
Create a new CBZ document containing the specified (parts of) pages from other documents.
- static void [CBZDocument](#) ([MuPDFContext](#) context, string fileName, params([MuPDFPage](#) page, [Rectangle](#) region, float zoom)[] pages)

- Create a new CBZ document containing the specified (parts of) pages from other documents.*
- static void `CBZDocument` (`MuPDFContext` context, string fileName, `IEnumerable< MuPDFPage >` pages, `CBZCreationOptions` options=default)
Create a new CBZ document containing the specified (parts of) pages from other documents.
 - static void `CBZDocument` (`MuPDFContext` context, string fileName, params `MuPDFPage[]` pages)
Create a new CBZ document containing the specified (parts of) pages from other documents.
 - static void `TextDocument` (`MuPDFContext` context, string fileName, `IEnumerable<(MuPDFPage page, Rectangle region, float zoom)>` pages, `TXTCreationOptions` options=default)
Create a new text document containing the specified (parts of) pages from other documents.
 - static void `TextDocument` (`MuPDFContext` context, string fileName, params(`MuPDFPage` page, `Rectangle` region, float zoom)[] pages)
Create a new text document containing the specified (parts of) pages from other documents.
 - static void `TextDocument` (`MuPDFContext` context, string fileName, `IEnumerable< MuPDFPage >` pages, `TXTCreationOptions` options=default)
Create a new text document containing the specified (parts of) pages from other documents.
 - static void `TextDocument` (`MuPDFContext` context, string fileName, params `MuPDFPage[]` pages)
Create a new text document containing the specified (parts of) pages from other documents.
 - static void `StructuredTextDocument` (`MuPDFContext` context, string fileName, `IEnumerable<(MuPDFPage page, Rectangle region, float zoom)>` pages, `TXTCreationOptions` options=default)
Create a new structured text XML document containing the specified (parts of) pages from other documents.
 - static void `StructuredTextDocument` (`MuPDFContext` context, string fileName, params(`MuPDFPage` page, `Rectangle` region, float zoom)[] pages)
Create a new structured text XML document containing the specified (parts of) pages from other documents.
 - static void `StructuredTextDocument` (`MuPDFContext` context, string fileName, `IEnumerable< MuPDFPage >` pages, `TXTCreationOptions` options=default)
Create a new structured text XML document containing the specified (parts of) pages from other documents.
 - static void `StructuredTextDocument` (`MuPDFContext` context, string fileName, params `MuPDFPage[]` pages)
Create a new structured text XML document containing the specified (parts of) pages from other documents.
 - static void `HTMLDocument` (`MuPDFContext` context, string fileName, `IEnumerable<(MuPDFPage page, Rectangle region, float zoom)>` pages, `HTMLCreationOptions` options=default)
Create a new HTML document containing the specified (parts of) pages from other documents.
 - static void `HTMLDocument` (`MuPDFContext` context, string fileName, params(`MuPDFPage` page, `Rectangle` region, float zoom)[] pages)
Create a new HTML document containing the specified (parts of) pages from other documents.
 - static void `HTMLDocument` (`MuPDFContext` context, string fileName, `IEnumerable< MuPDFPage >` pages, `HTMLCreationOptions` options=default)
Create a new HTML document containing the specified (parts of) pages from other documents.
 - static void `HTMLDocument` (`MuPDFContext` context, string fileName, params `MuPDFPage[]` pages)
Create a new HTML document containing the specified (parts of) pages from other documents.
 - static void `XHTMLDocument` (`MuPDFContext` context, string fileName, `IEnumerable<(MuPDFPage page, Rectangle region, float zoom)>` pages, `HTMLCreationOptions` options=default)
Create a new XHTML document containing the specified (parts of) pages from other documents.
 - static void `XHTMLDocument` (`MuPDFContext` context, string fileName, params(`MuPDFPage` page, `Rectangle` region, float zoom)[] pages)
Create a new XHTML document containing the specified (parts of) pages from other documents.
 - static void `XHTMLDocument` (`MuPDFContext` context, string fileName, `IEnumerable< MuPDFPage >` pages, `HTMLCreationOptions` options=default)
Create a new XHTML document containing the specified (parts of) pages from other documents.
 - static void `XHTMLDocument` (`MuPDFContext` context, string fileName, params `MuPDFPage[]` pages)
Create a new XHTML document containing the specified (parts of) pages from other documents.

7.4.1 Detailed Description

Contains methods to create documents by combining pages from other documents.

Definition at line 1205 of file [MuPDFDocument.Create.cs](#).

7.4.2 Member Function Documentation

7.4.2.1 CBZDocument() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.CBZDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable< MuPDFPage > pages,
    CBZCreationOptions options = default ) [static]
```

[Create](#) a new CBZ document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.2 CBZDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.CBZDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    CBZCreationOptions options = default ) [static]
```

[Create](#) a new CBZ document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1423 of file [MuPDFDocument.Create.cs](#).

7.4.2.3 CBZDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.CBZDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages) [static]
```

[Create](#) a new CBZ document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

7.4.2.4 CBZDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.CBZDocument (
    MuPDFContext context,
    string fileName,
    params (MuPDFPage page, Rectangle region, float zoom)[] pages) [static]
```

[Create](#) a new CBZ document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.5 Document() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.Document (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    IEnumerable< MuPDFPage > pages,
    bool includeAnnotations = true) [static]
```

[Create](#) a new document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.
<i>pages</i>	The pages to include in the document.

7.4.2.6 Document() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.Document (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    bool includeAnnotations = true ) [static]
```

[Create](#) a new document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.7 Document() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.Document (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    params MuPDFPage[] pages ) [static]
```

[Create](#) a new document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>pages</i>	The pages to include in the document.

7.4.2.8 Document() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.Document (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    params(MuPDFPage page, Rectangle region, float zoom)[] pages ) [static]
```

[Create](#) a new document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.9 HTMLDocument() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.HTMLDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable< MuPDFPage > pages,
    HTMLCreationOptions options = default ) [static]
```

[Create](#) a new HTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.10 HTMLDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.HTMLDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    HTMLCreationOptions options = default ) [static]
```

[Create](#) a new HTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1540 of file [MuPDFDocument.Create.cs](#).

7.4.2.11 HTMLDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.HTMLDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages ) [static]
```

[Create](#) a new HTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

7.4.2.12 HTMLDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.HTMLDocument (
    MuPDFContext context,
    string fileName,
    params (MuPDFPage page, Rectangle region, float zoom)[] pages ) [static]
```

[Create](#) a new HTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.13 PDFDocument() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.PDFDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable< MuPDFPage > pages,
    PDFCreationOptions options = default ) [static]
```

[Create](#) a new PDF document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.14 PDFDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.PDFDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    PDFCreationOptions options = default ) [static]
```

[Create](#) a new PDF document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1359 of file [MuPDFDocument.Create.cs](#).

7.4.2.15 PDFDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.PDFDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages ) [static]
```

[Create](#) a new PDF document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

7.4.2.16 PDFDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.PDFDocument (
    MuPDFContext context,
    string fileName,
    params(MuPDFPage page, Rectangle region, float zoom) [] pages ) [static]
```

[Create](#) a new PDF document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.17 StructuredTextDocument() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.StructuredTextDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable< MuPDFPage > pages,
    TXTCreationOptions options = default ) [static]
```

[Create](#) a new structured text XML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.18 StructuredTextDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.StructuredTextDocument (
```

```
MuPDFContext context,
string fileName,
IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
TXTCreationOptions options = default ) [static]
```

[Create](#) a new structured text XML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1501 of file [MuPDFDocument.Create.cs](#).

7.4.2.19 StructuredTextDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.StructuredTextDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages ) [static]
```

[Create](#) a new structured text XML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

7.4.2.20 StructuredTextDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.StructuredTextDocument (
    MuPDFContext context,
    string fileName,
    params (MuPDFPage page, Rectangle region, float zoom)[] pages ) [static]
```

[Create](#) a new structured text XML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<small>Generated by Doxygen</small>	

7.4.2.21 `SVGDocument()` [1/2]

```
static void MuPDFCore.MuPDFDocument.Create.SVGDocument (
    MuPDFContext context,
    string fileName,
    MuPDFPage page,
    Rectangle region,
    float zoom,
    SVGCreationOptions options = default ) [static]
```

[Create](#) a new SVG document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>page</i>	The page to include in the document.
<i>region</i>	The area of the page that should be included in the document
<i>zoom</i>	How much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1400 of file [MuPDFDocument.Create.cs](#).

7.4.2.22 `SVGDocument()` [2/2]

```
static void MuPDFCore.MuPDFDocument.Create.SVGDocument (
    MuPDFContext context,
    string fileName,
    MuPDFPage page,
    SVGCreationOptions options = default ) [static]
```

[Create](#) a new SVG document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>page</i>	The page to include in the document.
<i>options</i>	Options for the output format.

7.4.2.23 `TextDocument()` [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.TextDocument (
    MuPDFContext context,
```

```
    string fileName,
    IEnumerable< MuPDFPage > pages,
    TXTCreationOptions options = default ) [static]
```

[Create](#) a new text document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.24 TextDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.TextDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    TXTCreationOptions options = default ) [static]
```

[Create](#) a new text document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1462 of file [MuPDFDocument.Create.cs](#).

7.4.2.25 TextDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.TextDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages ) [static]
```

[Create](#) a new text document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

Generated by Doxygen

7.4.2.26 TextDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.TextDocument (
    MuPDFContext context,
    string fileName,
    params(MuPDFPage page, Rectangle region, float zoom)[] pages ) [static]
```

[Create](#) a new text document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.4.2.27 XHTMLDocument() [1/4]

```
static void MuPDFCore.MuPDFDocument.Create.XHTMLDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable< MuPDFPage > pages,
    HTMLCreationOptions options = default ) [static]
```

[Create](#) a new XHTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.
<i>options</i>	Options for the output format.

7.4.2.28 XHTMLDocument() [2/4]

```
static void MuPDFCore.MuPDFDocument.Create.XHTMLDocument (
    MuPDFContext context,
    string fileName,
    IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
    HTMLCreationOptions options = default ) [static]
```

[Create](#) a new XHTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.
<i>options</i>	Options for the output format.

Definition at line 1579 of file [MuPDFDocument.Create.cs](#).

7.4.2.29 XHTMLDocument() [3/4]

```
static void MuPDFCore.MuPDFDocument.Create.XHTMLDocument (
    MuPDFContext context,
    string fileName,
    params MuPDFPage[] pages) [static]
```

[Create](#) a new XHTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document.

7.4.2.30 XHTMLDocument() [4/4]

```
static void MuPDFCore.MuPDFDocument.Create.XHTMLDocument (
    MuPDFContext context,
    string fileName,
    params (MuPDFPage page, Rectangle region, float zoom)[] pages) [static]
```

[Create](#) a new XHTML document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

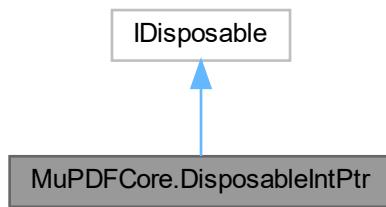
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFDocument.Create.cs

7.5 MuPDFCore.DisposableIntPtr Class Reference

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Inheritance diagram for MuPDFCore.DisposableIntPtr:



Public Member Functions

- [DisposableIntPtr \(IntPtr pointer\)](#)
Create a new [DisposableIntPtr](#).
- [DisposableIntPtr \(IntPtr pointer, long bytesAllocated\)](#)
Create a new [DisposableIntPtr](#), adding memory pressure to the GC to account for the allocation of unmanaged memory.
- void [Dispose \(\)](#)

7.5.1 Detailed Description

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Definition at line 523 of file [MuPDF.cs](#).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 DisposableIntPtr() [1/2]

```
MuPDFCore.DisposableIntPtr.DisposableIntPtr (
    IntPtr pointer )
```

Create a new [DisposableIntPtr](#).

Parameters

<i>pointer</i>	The pointer that should be freed upon disposing of this object.
----------------	---

Definition at line 539 of file [MuPDF.cs](#).

7.5.2.2 DisposableIntPtr() [2/2]

```
MuPDFCore.DisposableIntPtr.DisposableIntPtr (
    IntPtr pointer,
    long bytesAllocated )
```

Create a new [DisposableIntPtr](#), adding memory pressure to the GC to account for the allocation of unmanaged memory.

Parameters

<i>pointer</i>	The pointer that should be freed upon disposing of this object.
<i>bytesAllocated</i>	The number of bytes that have been allocated, for adding memory pressure.

Definition at line 549 of file [MuPDF.cs](#).

7.5.3 Member Function Documentation

7.5.3.1 Dispose()

```
void MuPDFCore.DisposableIntPtr.Dispose ( )
```

Definition at line 585 of file [MuPDF.cs](#).

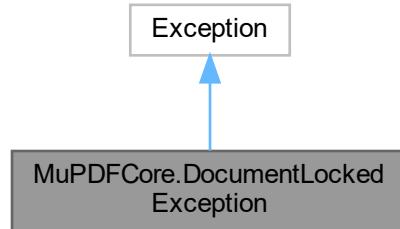
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

7.6 MuPDFCore.DocumentLockedException Class Reference

The exception that is thrown when an attempt is made to render an encrypted document without supplying the required password.

Inheritance diagram for MuPDFCore.DocumentLockedException:



7.6.1 Detailed Description

The exception that is thrown when an attempt is made to render an encrypted document without supplying the required password.

Definition at line 611 of file [MuPDF.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

7.7 MuPDFCore.PDFCreationOptions.DocumentPermissions Class Reference

Specifies which actions can be performed without the owner password.

Public Types

- enum [PrintingPermission](#)
Quality of allowed printing operations.
- enum [ExtractionPermission](#)
Allowed text and graphics copy/extraction operations.
- enum [FormPermission](#)
Allowed form and annotation operations.

Properties

- **PrintingPermission Printing** = PrintingPermission.HighFidelity [get, set]
Specifies if printing is allowed without the owner password.
- **ExtractionPermission Extraction** = ExtractionPermission.CopyAndExtractTextAndGraphics [get, set]
Specifies whether text can be extracted without the owner password.
- **bool AllowDocumentRestructuring** [get, set]
Specifies whether the document can be restructured without the owner password (e.g., by inserting, rotating or deleting pages, bookmarks and thumbnail images).
- **FormPermission Annotations** [get, set]
Specifies whether forms can be filled-in and annotations can be added to the document without the owner password.
- **bool AllowDocumentModification** [get, set]
Specifies whether the document can be modified without the owner password.

7.7.1 Detailed Description

Specifies which actions can be performed without the owner password.

Definition at line 106 of file [MuPDFDocument.Create.cs](#).

7.7.2 Member Enumeration Documentation

7.7.2.1 ExtractionPermission

```
enum MuPDFCore.PDFCreationOptions.DocumentPermissions.ExtractionPermission
```

Allowed text and graphics copy/extraction operations.

Definition at line 132 of file [MuPDFDocument.Create.cs](#).

7.7.2.2 FormPermission

```
enum MuPDFCore.PDFCreationOptions.DocumentPermissions.FormPermission
```

Allowed form and annotation operations.

Definition at line 153 of file [MuPDFDocument.Create.cs](#).

7.7.2.3 PrintingPermission

```
enum MuPDFCore.PDFCreationOptions.DocumentPermissions.PrintingPermission
```

Quality of allowed printing operations.

Definition at line 111 of file [MuPDFDocument.Create.cs](#).

7.7.3 Property Documentation

7.7.3.1 AllowDocumentModification

```
bool MuPDFCore.PDFCreationOptions.DocumentPermissions.AllowDocumentModification [get], [set]
```

Specifies whether the document can be modified without the owner password.

Definition at line 194 of file [MuPDFDocument.Create.cs](#).

7.7.3.2 AllowDocumentRestructuring

```
bool MuPDFCore.PDFCreationOptions.DocumentPermissions.AllowDocumentRestructuring [get], [set]
```

Specifies whether the document can be restructured without the owner password (e.g., by inserting, rotating or deleting pages, bookmarks and thumbnail images).

Definition at line 184 of file [MuPDFDocument.Create.cs](#).

7.7.3.3 Annotations

```
FormPermission MuPDFCore.PDFCreationOptions.DocumentPermissions.Annotations [get], [set]
```

Specifies whether forms can be filled-in and annotations can be added to the document without the owner password.

Definition at line 189 of file [MuPDFDocument.Create.cs](#).

7.7.3.4 Extraction

```
ExtractionPermission MuPDFCore.PDFCreationOptions.DocumentPermissions.Extraction = Extraction←  
Permission.CopyAndExtractTextAndGraphics [get], [set]
```

Specifies whether text can be extracted without the owner password.

Definition at line 179 of file [MuPDFDocument.Create.cs](#).

7.7.3.5 Printing

```
PrintingPermission MuPDFCore.PDFCreationOptions.DocumentPermissions.Printing = Printing←  
Permission.HighFidelity [get], [set]
```

Specifies if printing is allowed without the owner password.

Definition at line 174 of file [MuPDFDocument.Create.cs](#).

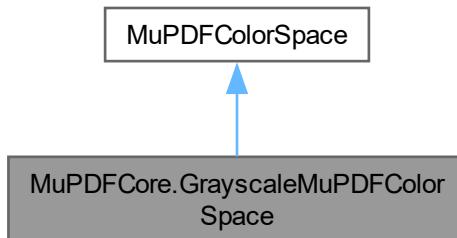
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFDocument.Create.cs](#)

7.8 MuPDFCore.GrayscaleMuPDFColorSpace Class Reference

Represents a colour space where each pixel is stored as a single byte.

Inheritance diagram for MuPDFCore.GrayscaleMuPDFColorSpace:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]

Additional Inherited Members

7.8.1 Detailed Description

Represents a colour space where each pixel is stored as a single byte.

Definition at line 239 of file [MuPDFColorSpace.cs](#).

7.8.2 Property Documentation

7.8.2.1 NumBytes

```
override int MuPDFCore.GrayscaleMuPDFColorSpace.NumBytes [get]
```

Definition at line 245 of file [MuPDFColorSpace.cs](#).

7.8.2.2 Type

```
override ColorSpaceType MuPDFCore.GrayscaleMuPDFColorSpace.Type [get]
```

Definition at line 242 of file [MuPDFColorSpace.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFColorSpace.cs

7.9 MuPDFCore.HTMLCreationOptions Class Reference

Options for creating an HTML or XHTML document.

Properties

- bool `IncludeAnnotations` = true [get, set]
If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.
- bool `InhibitSpaces` = false [get, set]
Do not add spaces between gaps in the text.
- bool `PreserveImages` = false [get, set]
Keep images in the output.
- bool `PreserveLigatures` = false [get, set]
Do not expand ligatures into constituent characters.
- bool `PreserveWhitespace` = false [get, set]
Do not convert all whitespace into space characters.
- bool `PreserveSpans` = false [get, set]
Do not merge spans on the same line.
- bool `Dehyphenate` = false [get, set]
Attempt to join up hyphenated words.
- bool `UseCIDForUnknownUnicode` = false [get, set]
Guess unicode from CID if normal mapping fails.
- bool `IncludeCharactersOutsideMediaBox` = false [get, set]
Include characters that are outside of the page's mediabox.

7.9.1 Detailed Description

Options for creating an HTML or XHTML document.

Definition at line 1050 of file [MuPDFDocument.Create.cs](#).

7.9.2 Property Documentation

7.9.2.1 Dehyphenate

```
bool MuPDFCore.HTMLCreationOptions.Dehyphenate = false [get], [set]
```

Attempt to join up hyphenated words.

Definition at line 1085 of file [MuPDFDocument.Create.cs](#).

7.9.2.2 IncludeAnnotations

```
bool MuPDFCore.HTMLCreationOptions.IncludeAnnotations = true [get], [set]
```

If this is `true`, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.

Definition at line 1055 of file [MuPDFDocument.Create.cs](#).

7.9.2.3 IncludeCharactersOutsideMediaBox

```
bool MuPDFCore.HTMLCreationOptions.IncludeCharactersOutsideMediaBox = false [get], [set]
```

Include characters that are outside of the page's mediabox.

Definition at line 1095 of file [MuPDFDocument.Create.cs](#).

7.9.2.4 InhibitSpaces

```
bool MuPDFCore.HTMLCreationOptions.InhibitSpaces = false [get], [set]
```

Do not add spaces between gaps in the text.

Definition at line 1060 of file [MuPDFDocument.Create.cs](#).

7.9.2.5 **PreserveImages**

```
bool MuPDFCore.HTMLCreationOptions.PreserveImages = false [get], [set]
```

Keep images in the output.

Definition at line 1065 of file [MuPDFDocument.Create.cs](#).

7.9.2.6 **PreserveLigatures**

```
bool MuPDFCore.HTMLCreationOptions.PreserveLigatures = false [get], [set]
```

Do not expand ligatures into constituent characters.

Definition at line 1070 of file [MuPDFDocument.Create.cs](#).

7.9.2.7 **PreserveSpans**

```
bool MuPDFCore.HTMLCreationOptions.PreserveSpans = false [get], [set]
```

Do not merge spans on the same line.

Definition at line 1080 of file [MuPDFDocument.Create.cs](#).

7.9.2.8 **PreserveWhitespace**

```
bool MuPDFCore.HTMLCreationOptions.PreserveWhitespace = false [get], [set]
```

Do not convert all whitespace into space characters.

Definition at line 1075 of file [MuPDFDocument.Create.cs](#).

7.9.2.9 **UseCIDForUnknownUnicode**

```
bool MuPDFCore.HTMLCreationOptions.UseCIDForUnknownUnicode = false [get], [set]
```

Guess unicode from CID if normal mapping fails.

Definition at line 1090 of file [MuPDFDocument.Create.cs](#).

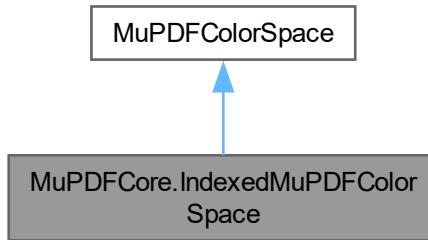
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFDocument.Create.cs](#)

7.10 MuPDFCore.IndexedMuPDFColorSpace Class Reference

Represents a colour space where each pixel is stored as a single byte mapping the pixel to a colour in the lookup table.

Inheritance diagram for MuPDFCore.IndexedMuPDFColorSpace:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]
- byte[] [LookupTable](#) [get]
The lookup table containing the colour index.
- [MuPDFColorSpace BaseColorSpace](#) [get]
The colour space in which the colours in the [LookupTable](#) are expressed.

Additional Inherited Members

7.10.1 Detailed Description

Represents a colour space where each pixel is stored as a single byte mapping the pixel to a colour in the lookup table.

Definition at line 324 of file [MuPDFColorSpace.cs](#).

7.10.2 Property Documentation

7.10.2.1 BaseColorSpace

[MuPDFColorSpace](#) [MuPDFCore.IndexedMuPDFColorSpace.BaseColorSpace](#) [get]

The colour space in which the colours in the [LookupTable](#) are expressed.

Definition at line 340 of file [MuPDFColorSpace.cs](#).

7.10.2.2 `LookupTable`

```
byte [ ] MuPDFCore.IndexedMuPDFColorSpace.LookupTable [get]
```

The lookup table containing the colour index.

Definition at line 335 of file [MuPDFColorSpace.cs](#).

7.10.2.3 `NumBytes`

```
override int MuPDFCore.IndexedMuPDFColorSpace.NumBytes [get]
```

Definition at line 330 of file [MuPDFColorSpace.cs](#).

7.10.2.4 `Type`

```
override ColorSpaceType MuPDFCore.IndexedMuPDFColorSpace.Type [get]
```

Definition at line 327 of file [MuPDFColorSpace.cs](#).

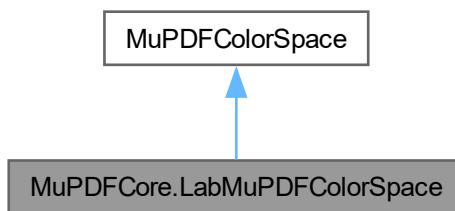
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFColorSpace.cs](#)

7.11 `MuPDFCore.LabMuPDFColorSpace` Class Reference

Represents a colour space where each pixel is stored as three bytes encoding the L*, a, and b components.

Inheritance diagram for `MuPDFCore.LabMuPDFColorSpace`:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]

Additional Inherited Members

7.11.1 Detailed Description

Represents a colour space where each pixel is stored as three bytes encoding the L*, a, and b components.

Definition at line [290](#) of file [MuPDFColorSpace.cs](#).

7.11.2 Property Documentation

7.11.2.1 NumBytes

```
override int MuPDFCore.LabMuPDFColorSpace.NumBytes [get]
```

Definition at line [296](#) of file [MuPDFColorSpace.cs](#).

7.11.2.2 Type

```
override ColorSpaceType MuPDFCore.LabMuPDFColorSpace.Type [get]
```

Definition at line [293](#) of file [MuPDFColorSpace.cs](#).

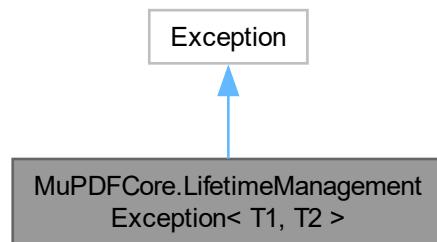
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFColorSpace.cs](#)

7.12 MuPDFCore.LifetimeManagementException< T1, T2 > Class Template Reference

The exception that is thrown when the lifetime of disposable objects is not managed properly.

Inheritance diagram for MuPDFCore.LifetimeManagementException< T1, T2 >:



Public Member Functions

- `LifetimeManagementException (LifetimeManagementException< T1, T2 > baseException, string appendedMessage)`
Create a new LifetimeManagementException<T1, T2> by appending the specified message to the original exception message.

Properties

- override string `Message [get]`
- `T1 Disposing [get]`
The object whose disposal caused the exception.
- `T2 Owner [get]`
The object that had already been disposed.

7.12.1 Detailed Description

The exception that is thrown when the lifetime of disposable objects is not managed properly.

Definition at line 619 of file [MuPDF.cs](#).

7.12.2 Constructor & Destructor Documentation

7.12.2.1 LifetimeManagementException()

```
MuPDFCore.LifetimeManagementException< T1, T2 >.LifetimeManagementException (
    LifetimeManagementException< T1, T2 > baseException,
    string appendedMessage )
```

Create a new LifetimeManagementException<T1, T2> by appending the specified message to the original exception message.

Parameters

<code>baseException</code>	The original LifetimeManagementException<T1, T2>.
<code>appendedMessage</code>	The message to be appended to the original exception message.

Definition at line 651 of file [MuPDF.cs](#).

7.12.3 Property Documentation

7.12.3.1 Disposing

```
T1 MuPDFCore.LifetimeManagementException< T1, T2 >.Disposing [get]
```

The object whose disposal caused the exception.

Definition at line 627 of file [MuPDF.cs](#).

7.12.3.2 Message

```
override string MuPDFCore.LifetimeManagementException< T1, T2 >.Message [get]
```

Definition at line 622 of file [MuPDF.cs](#).

7.12.3.3 Owner

```
T2 MuPDFCore.LifetimeManagementException< T1, T2 >.Owner [get]
```

The object that had already been disposed.

Definition at line 632 of file [MuPDF.cs](#).

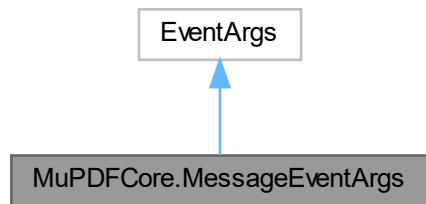
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDF.cs](#)

7.13 MuPDFCore.MessageEventArgs Class Reference

EventArgs for the [MuPDF.StandardOutputMessage](#) and [MuPDF.StandardErrorMessage](#) events.

Inheritance diagram for MuPDFCore.MessageEventArgs:



Public Member Functions

- [MessageEventArgs](#) (string message)
Create a new [MessageEventArgs](#) instance.

Properties

- string [Message](#) [get]
The message that has been logged.

7.13.1 Detailed Description

EventArgs for the [MuPDF.StandardOutputMessage](#) and [MuPDF.StandardErrorMessage](#) events.

Definition at line [708](#) of file [MuPDF.cs](#).

7.13.2 Constructor & Destructor Documentation

7.13.2.1 MessageEventArgs()

```
MuPDFCore.MessageEventArgs.MessageEventArgs (
    string message )
```

Create a new [MessageEventArgs](#) instance.

Parameters

<code>message</code>	The message that has been logged.
----------------------	-----------------------------------

Definition at line [719](#) of file [MuPDF.cs](#).

7.13.3 Property Documentation

7.13.3.1 Message

```
string MuPDFCore.MessageEventArgs.Message [get]
```

The message that has been logged.

Definition at line [713](#) of file [MuPDF.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDF.cs](#)

7.14 MuPDFCore.MuPDF Class Reference

Contains static methods to perform setup operations.

Static Public Member Functions

- static async Task [RedirectOutput \(\)](#)

Redirects output messages from the native MuPDF library to the StandardOutputMessage and StandardErrorMessage events. Note that this has side-effects.

- static void [ResetOutput \(\)](#)

Reset the default standard output and error streams for the native MuPDF library.

Events

- static EventHandler< [MessageEventArgs](#) > [StandardOutputMessage](#)

This event is invoked when RedirectOutput has been called and the native MuPDF library writes to the standard output stream.

- static EventHandler< [MessageEventArgs](#) > [StandardErrorMessage](#)

This event is invoked when RedirectOutput has been called and the native MuPDF library writes to the standard error stream.

7.14.1 Detailed Description

Contains static methods to perform setup operations.

Definition at line [728](#) of file [MuPDF.cs](#).

7.14.2 Member Function Documentation

7.14.2.1 [RedirectOutput\(\)](#)

```
static async Task MuPDFCore.MuPDF.RedirectOutput ( ) [static]
```

Redirects output messages from the native MuPDF library to the StandardOutputMessage and StandardErrorMessage events. Note that this has side-effects.

Returns

A Task that finishes when the output streams have been redirected.

Definition at line [757](#) of file [MuPDF.cs](#).

7.14.2.2 ResetOutput()

```
static void MuPDFCore.MuPDF.ResetOutput ( ) [static]
```

Reset the default standard output and error streams for the native [MuPDF](#) library.

Definition at line [978](#) of file [MuPDF.cs](#).

7.14.3 Event Documentation

7.14.3.1 StandardErrorMessage

```
EventHandler<MessageEventArgs> MuPDFCore.MuPDF.StandardErrorMessage [static]
```

This event is invoked when [RedirectOutput](#) has been called and the native [MuPDF](#) library writes to the standard error stream.

Definition at line [751](#) of file [MuPDF.cs](#).

7.14.3.2 StandardOutputMessage

```
EventHandler<MessageEventArgs> MuPDFCore.MuPDF.StandardOutputMessage [static]
```

This event is invoked when [RedirectOutput](#) has been called and the native [MuPDF](#) library writes to the standard output stream.

Definition at line [746](#) of file [MuPDF.cs](#).

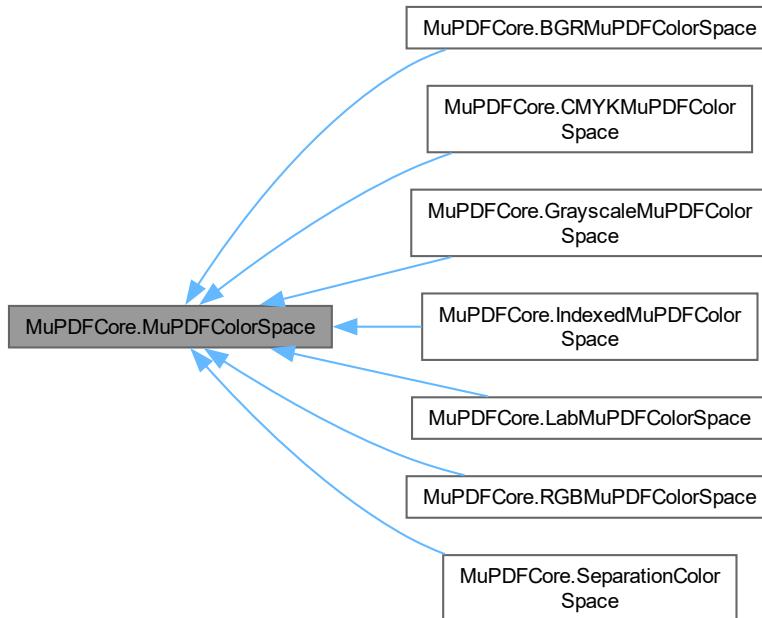
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

7.15 MuPDFCore.MuPDFColorSpace Class Reference

Represents a colour space.

Inheritance diagram for MuPDFCore.MuPDFColorSpace:



Public Member Functions

- override string **ToString ()**

Properties

- abstract **ColorSpaceType Type [get]**
The colour space type.
- abstract int **NumBytes [get]**
The number of bytes necessary to represent a single pixel in this colour space.
- virtual string **Name [get]**
The name of the colour space.
- virtual **MuPDFColorSpace RootColorSpace [get]**
The final colour space in which all pixels will eventually be represented.

7.15.1 Detailed Description

Represents a colour space.

Definition at line 73 of file [MuPDFColorSpace.cs](#).

7.15.2 Property Documentation

7.15.2.1 Name

```
virtual string MuPDFCore.MuPDFColorSpace.Name [get]
```

The name of the colour space.

Definition at line [88](#) of file [MuPDFColorSpace.cs](#).

7.15.2.2 NumBytes

```
abstract int MuPDFCore.MuPDFColorSpace.NumBytes [get]
```

The number of bytes necessary to represent a single pixel in this colour space.

Definition at line [83](#) of file [MuPDFColorSpace.cs](#).

7.15.2.3 RootColorSpace

```
virtual MuPDFColorSpace MuPDFCore.MuPDFColorSpace.RootColorSpace [get]
```

The final colour space in which all pixels will eventually be represented.

Definition at line [102](#) of file [MuPDFColorSpace.cs](#).

7.15.2.4 Type

```
abstract ColorSpaceType MuPDFCore.MuPDFColorSpace.Type [get]
```

The colour space type.

Definition at line [78](#) of file [MuPDFColorSpace.cs](#).

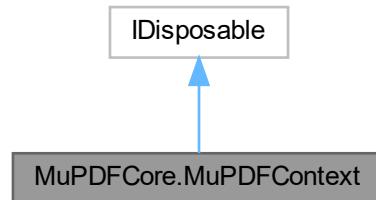
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFColorSpace.cs

7.16 MuPDFCore.MuPDFContext Class Reference

A wrapper around a [MuPDF](#) context object, which contains the exception stack and the resource cache store.

Inheritance diagram for MuPDFCore.MuPDFContext:



Public Member Functions

- [MuPDFContext](#) (uint storeSize=256<< 20)
Create a new [MuPDFContext](#) instance with the specified cache store size.
- void [ClearStore](#) ()
Evict all items from the resource cache store (freeing the memory where they were held).
- void [ShrinkStore](#) (double fraction)
Evict items from the resource cache store (freeing the memory where they were held) until the the size of the store drops to the specified fraction of the current size.
- void [Dispose](#) ()

Properties

- long [StoreSize](#) [get]
The current size in bytes of the resource cache store. Read-only.
- long [StoreMaxSize](#) [get]
The maximum size in bytes of the resource cache store. Read-only.
- int [AntiAliasing](#) [set]
Sets the current anti-aliasing level. Changing this value will affect both the [TextAntiAliasing](#) and the [GraphicsAntiAliasing](#).
- int [TextAntiAliasing](#) [get, set]
Gets or sets the current text anti-aliasing level.
- int [GraphicsAntiAliasing](#) [get, set]
Gets or sets the current graphics anti-aliasing level.

7.16.1 Detailed Description

A wrapper around a [MuPDF](#) context object, which contains the exception stack and the resource cache store.

Definition at line 26 of file [MuPDFContext.cs](#).

7.16.2 Constructor & Destructor Documentation

7.16.2.1 MuPDFContext()

```
MuPDFCore.MuPDFContext.MuPDFContext (
    uint storeSize = 256 << 20 )
```

Create a new [MuPDFContext](#) instance with the specified cache store size.

Parameters

<code>storeSize</code>	The maximum size in bytes of the resource cache store. The default value is 256 MiB.
------------------------	--

Definition at line 151 of file [MuPDFContext.cs](#).

7.16.3 Member Function Documentation

7.16.3.1 ClearStore()

```
void MuPDFCore.MuPDFContext.ClearStore ( )
```

Evict all items from the resource cache store (freeing the memory where they were held).

Definition at line 182 of file [MuPDFContext.cs](#).

7.16.3.2 Dispose()

```
void MuPDFCore.MuPDFContext.Dispose ( )
```

Definition at line 258 of file [MuPDFContext.cs](#).

7.16.3.3 ShrinkStore()

```
void MuPDFCore.MuPDFContext.ShrinkStore (
    double fraction )
```

Evict items from the resource cache store (freeing the memory where they were held) until the the size of the store drops to the specified fraction of the current size.

Parameters

<i>fraction</i>	The fraction of the current size that constitutes the target size of the store. If this is ≤ 0 , the cache is cleared. If this is ≥ 1 , nothing happens.
-----------------	--

Definition at line 191 of file [MuPDFContext.cs](#).

7.16.4 Property Documentation

7.16.4.1 AntiAliasing

```
int MuPDFCore.MuPDFContext.AntiAliasing [set]
```

Sets the current anti-aliasing level. Changing this value will affect both the [TextAntiAliasing](#) and the [GraphicsAntiAliasing](#).

Definition at line 89 of file [MuPDFContext.cs](#).

7.16.4.2 GraphicsAntiAliasing

```
int MuPDFCore.MuPDFContext.GraphicsAntiAliasing [get], [set]
```

Gets or sets the current graphics anti-aliasing level.

Definition at line 127 of file [MuPDFContext.cs](#).

7.16.4.3 StoreMaxSize

```
long MuPDFCore.MuPDFContext.StoreMaxSize [get]
```

The maximum size in bytes of the resource cache store. Read-only.

Definition at line 77 of file [MuPDFContext.cs](#).

7.16.4.4 StoreSize

```
long MuPDFCore.MuPDFContext.StoreSize [get]
```

The current size in bytes of the resource cache store. Read-only.

Definition at line 41 of file [MuPDFContext.cs](#).

7.16.4.5 TextAntiAliasing

```
int MuPDFCore.MuPDFContext.TextAntiAliasing [get], [set]
```

Gets or sets the current text anti-aliasing level.

Definition at line 105 of file [MuPDFContext.cs](#).

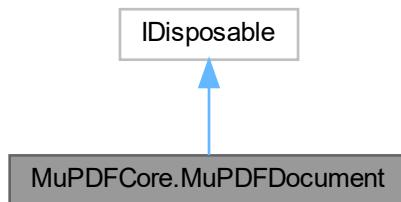
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFContext.cs](#)

7.17 MuPDFCore.MuPDFDocument Class Reference

A wrapper over a [MuPDF](#) document object, which contains possibly multiple pages.

Inheritance diagram for MuPDFCore.MuPDFDocument:



Classes

- class [Create](#)
Contains methods to create documents by combining pages from other documents.

Public Member Functions

- [MuPDFDocument \(MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes fileType\)](#)
Create a new [MuPDFDocument](#) from data bytes accessible through the specified pointer.
- [MuPDFDocument \(MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes fileType, ref IDisposable dataHolder\)](#)
Create a new [MuPDFDocument](#) from data bytes accessible through the specified pointer.
- [MuPDFDocument \(MuPDFContext context, byte\[\] data, InputFileTypes fileType\)](#)
Create a new [MuPDFDocument](#) from an array of bytes.
- [MuPDFDocument \(MuPDFContext context, ref MemoryStream data, InputFileTypes fileType\)](#)
Create a new [MuPDFDocument](#) from a [MemoryStream](#).
- [MuPDFDocument \(MuPDFContext context, string fileName\)](#)

- Create a new [MuPDFDocument](#) from a file.*
- void [ClearCache \(\)](#)
Discard all the display lists that have been loaded from the document, possibly freeing some memory in the case of a huge document.
 - void [Layout \(float width, float height, float em\)](#)
Sets the document layout for reflowable document types (e.g., HTML, MOBI). Does not have any effect for documents with a fixed layout (e.g., PDF).
 - void [LayoutSinglePage \(float width, float em\)](#)
Sets the document layout for reflowable document types (e.g., HTML, MOBI), so that the document is rendered to a single page, as tall as necessary. Does not have any effect for documents with a fixed layout (e.g., PDF).
 - byte[] [Render \(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, bool includeAnnotations=true\)](#)
Render (part of) a page to an array of bytes.
 - byte[] [Render \(int pageNumber, double zoom, PixelFormats pixelFormat, bool includeAnnotations=true\)](#)
Render a page to an array of bytes.
 - void [Render \(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, IntPtr destination, bool includeAnnotations=true\)](#)
Render (part of) a page to the specified destination.
 - void [Render \(int pageNumber, double zoom, PixelFormats pixelFormat, IntPtr destination, bool includeAnnotations=true\)](#)
Render a page to the specified destination.
 - Span< byte > [Render \(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, out IDisposable disposable, bool includeAnnotations=true\)](#)
Render (part of) a page to a Span<byte>.
 - Span< byte > [Render \(int pageNumber, double zoom, PixelFormats pixelFormat, out IDisposable disposable, bool includeAnnotations=true\)](#)
Render a page to a Span<byte>.
 - [MuPDFMultiThreadedPageRenderer GetMultiThreadedRenderer \(int pageNumber, int threadCount, bool includeAnnotations=true\)](#)
Create a new [MuPDFMultiThreadedPageRenderer](#) that renders the specified page with the specified number of threads.
 - int [GetRenderedSize \(int pageNumber, double zoom, PixelFormats pixelFormat\)](#)
Determine how many bytes will be necessary to render the specified page at the specified zoom level, using the the specified pixel format.
 - void [SaveImage \(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, string fileName, RasterOutputFileTypes fileType, bool includeAnnotations=true\)](#)
Save (part of) a page to an image file in the specified format.
 - void [SaveImageAsJPEG \(int pageNumber, Rectangle region, double zoom, string fileName, int quality, bool includeAnnotations=true\)](#)
Save (part of) a page to an image file in JPEG format, with the specified quality.
 - void [SaveImage \(int pageNumber, double zoom, PixelFormats pixelFormat, string fileName, RasterOutputFileTypes fileType, bool includeAnnotations=true\)](#)
Save a page to an image file in the specified format.
 - void [SaveImageAsJPEG \(int pageNumber, double zoom, string fileName, int quality, bool includeAnnotations=true\)](#)
Save a page to an image file in JPEG format, with the specified quality.
 - void [WriteImage \(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, Stream outputStream, RasterOutputFileTypes fileType, bool includeAnnotations=true\)](#)
Write (part of) a page to an image stream in the specified format.
 - void [WriteImageAsJPEG \(int pageNumber, Rectangle region, double zoom, Stream outputStream, int quality, bool includeAnnotations=true\)](#)
Write (part of) a page to an image stream in JPEG format, with the specified quality.
 - void [WriteImage \(int pageNumber, double zoom, PixelFormats pixelFormat, Stream outputStream, RasterOutputFileTypes fileType, bool includeAnnotations=true\)](#)
Write a page to an image stream in JPEG format, with the specified quality.

Write a page to an image stream in the specified format.

- void [WriteImageAsJPEG](#) (int pageNumber, double zoom, Stream outputStream, int quality, bool includeAnnotations=true)

Write a page to an image stream in JPEG format, with the specified quality.

- MuPDFStructuredTextPage [GetStructuredTextPage](#) (int pageNumber, bool includeAnnotations=true, StructuredTextFlags flags=StructuredTextFlags.None)

Creates a new MuPDFStructuredTextPage from the specified page. This contains information about the text layout that can be used for highlighting and searching. The reading order is taken from the order the text is drawn in the source file, so may not be accurate.

- MuPDFStructuredTextPage [GetStructuredTextPage](#) (int pageNumber, TesseractLanguage ocrLanguage, bool includeAnnotations=true, StructuredTextFlags flags=StructuredTextFlags.None, CancellationToken cancellationToken=default, IProgress<OCRProgressInfo> progress=null)

Creates a new MuPDFStructuredTextPage from the specified page, using optical character recognition (OCR) to determine what text is written on the image. This contains information about the text layout that can be used for highlighting and searching.

- async Task<MuPDFStructuredTextPage> [GetStructuredTextPageAsync](#) (int pageNumber, TesseractLanguage ocrLanguage, bool includeAnnotations=true, StructuredTextFlags flags=StructuredTextFlags.None, CancellationToken cancellationToken=default, IProgress<OCRProgressInfo> progress=null)

Creates a new MuPDFStructuredTextPage from the specified page, using optical character recognition (OCR) to determine what text is written on the image. This contains information about the text layout that can be used for highlighting and searching. The OCR step is run asynchronously, e.g. to avoid blocking the UI thread.

- string [ExtractText](#) (string separator=null, bool includeAnnotations=true)

Extracts all the text from the document and returns it as a string. The reading order is taken from the order the text is drawn in the source file, so may not be accurate.

- string [ExtractText](#) (TesseractLanguage ocrLanguage, string separator=null, bool includeAnnotations=true)

Extracts all the text from the document and returns it as a string, using optical character recognition (OCR) to determine what text is written on the image.

- async Task<string> [ExtractTextAsync](#) (TesseractLanguage ocrLanguage, string separator=null, bool includeAnnotations=true, CancellationToken cancellationToken=default, IProgress<OCRProgressInfo> progress=null)

Extracts all the text from the document and returns it as a string, using optical character recognition (OCR) to determine what text is written on the image. The OCR step is run asynchronously, e.g. to avoid blocking the UI thread.

- bool [TryUnlock](#) (string password)

Attempts to unlock the document with the supplied password.

- bool [TryUnlock](#) (string password, out PasswordTypes passwordType)

Attempts to unlock the document with the supplied password.

- void [Dispose](#) ()

Static Public Member Functions

- static void [CreateDocument](#) (MuPDFContext context, string fileName, DocumentOutputFileTypes fileType, bool includeAnnotations=true, params(MuPDFPage page, Rectangle region, float zoom)[] pages)

Create a new document containing the specified (parts of) pages from other documents.

- static void [CreateDocument](#) (MuPDFContext context, string fileName, DocumentOutputFileTypes fileType, bool includeAnnotations=true, params MuPDFPage[] pages)

Create a new document containing the specified pages from other documents.

- static int [GetRenderedSize](#) (Rectangle region, double zoom, PixelFormats pixelFormat)

Determine how many bytes will be necessary to render the specified region in page units at the specified zoom level, using the the specified pixel format.

Properties

- **MuPDFPageCollection Pages** [get]
The pages contained in the document.
- **bool ClipToPageBounds = true** [get, set]
Defines whether the images resulting from rendering operations should be clipped to the page boundaries.
- **EncryptionState EncryptionState** [get]
Describes the encryption state of the document.
- **RestrictionState RestrictionState** [get]
Describes the restriction state of the document.
- **DocumentRestrictions Restrictions** [get]
Describes the operations that are restricted on the document. This is not actually enforced by the library, but library users should only allow these operations if the document has been unlocked with the owner password (i.e. if `RestrictionState` is `RestrictionState.Unlocked`).
- **MuPDFOutline Outline** [get]
The document outline (table of contents). If this document does not have an outline, this object will be empty, but not null. The outline is loaded from the document at the first access.
- **MuPDFOptionalContentGroupData OptionalContentGroupData** [get]
Contains information about optional content groups (aka layers). If this document does not contain any optional content groups, this object will be null. The optional content group data is loaded from the document at the first access.

7.17.1 Detailed Description

A wrapper over a [MuPDF](#) document object, which contains possibly multiple pages.

Definition at line 1178 of file [MuPDFDocument.Create.cs](#).

7.17.2 Constructor & Destructor Documentation

7.17.2.1 MuPDFDocument() [1/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
    MuPDFContext context,
    IntPtr dataAddress,
    long dataLength,
    InputFileTypes fileType )
```

Create a new [MuPDFDocument](#) from data bytes accessible through the specified pointer.

Parameters

<code>context</code>	The context that will own this document.
<code>dataAddress</code>	A pointer to the data bytes that make up the document.
<code>dataLength</code>	The number of bytes to read from the specified address.
<code>fileType</code>	The type of the document to read.

Definition at line 189 of file [MuPDFDocument.cs](#).

7.17.2.2 MuPDFDocument() [2/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
    MuPDFContext context,
    IntPtr dataAddress,
    long dataLength,
    InputFileTypes fileType,
    ref IDisposable dataHolder )
```

Create a new [MuPDFDocument](#) from data bytes accessible through the specified pointer.

Parameters

<code>context</code>	The context that will own this document.
<code>dataAddress</code>	A pointer to the data bytes that make up the document.
<code>dataLength</code>	The number of bytes to read from the specified address.
<code>fileType</code>	The type of the document to read.
<code>dataHolder</code>	An IDisposable that will be disposed when the MuPDFDocument is disposed.

Definition at line 199 of file [MuPDFDocument.cs](#).

7.17.2.3 MuPDFDocument() [3/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
    MuPDFContext context,
    byte[] data,
    InputFileTypes fileType )
```

Create a new [MuPDFDocument](#) from an array of bytes.

Parameters

<code>context</code>	The context that will own this document.
<code>data</code>	An array containing the data bytes that make up the document. This must not be altered until after the MuPDFDocument has been disposed! The address of the array will be pinned, which may cause degradation in the Garbage Collector's performance, and is thus only advised for short-lived documents. To avoid this issue, marshal the bytes to unmanaged memory and use one of the IntPtr constructors.
<code>fileType</code>	The type of the document to read.

Definition at line 314 of file [MuPDFDocument.cs](#).

7.17.2.4 MuPDFDocument() [4/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
    MuPDFContext context,
    ref MemoryStream data,
    InputFileTypes fileType )
```

Create a new MuPDFDocument from a MemoryStream.

Parameters

<i>context</i>	The context that will own this document.
<i>data</i>	The MemoryStream containing the data that makes up the document. This will be disposed when the MuPDFDocument has been disposed and must not be disposed externally! The address of the MemoryStream's buffer will be pinned, which may cause degradation in the Garbage Collector's performance, and is thus only advised for short-lived documents. To avoid this issue, marshal the bytes to unmanaged memory and use one of the IntPtr constructors.
<i>fileType</i>	The type of the document to read.

Definition at line 431 of file [MuPDFDocument.cs](#).

7.17.2.5 MuPDFDocument() [5/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
    MuPDFContext context,
    string fileName )
```

Create a new MuPDFDocument from a file.

Parameters

<i>context</i>	The context that will own this document.
<i>fileName</i>	The path to the file to open.

Definition at line 552 of file [MuPDFDocument.cs](#).

7.17.3 Member Function Documentation

7.17.3.1 ClearCache()

```
void MuPDFCore.MuPDFDocument.ClearCache ( )
```

Discard all the display lists that have been loaded from the document, possibly freeing some memory in the case of a huge document.

Definition at line 698 of file [MuPDFDocument.cs](#).

7.17.3.2 CreateDocument() [1/2]

```
static void MuPDFCore.MuPDFDocument.CreateDocument (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    bool includeAnnotations = true,
    params MuPDFPage[] pages) [static]
```

[Create](#) a new document containing the specified pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>pages</i>	The pages to include in the document.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

7.17.3.3 CreateDocument() [2/2]

```
static void MuPDFCore.MuPDFDocument.CreateDocument (
    MuPDFContext context,
    string fileName,
    DocumentOutputFileTypes fileType,
    bool includeAnnotations = true,
    params (MuPDFPage page, Rectangle region, float zoom)[] pages) [static]
```

[Create](#) a new document containing the specified (parts of) pages from other documents.

Parameters

<i>context</i>	The context that was used to open the documents.
<i>fileName</i>	The output file name.
<i>fileType</i>	The output file format.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.
<i>pages</i>	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

7.17.3.4 Dispose()

```
void MuPDFCore.MuPDFDocument.Dispose ()
```

Definition at line 1738 of file [MuPDFDocument.cs](#).

7.17.3.5 ExtractText() [1/2]

```
string MuPDFCore.MuPDFDocument.ExtractText (
    string separator = null,
    bool includeAnnotations = true )
```

Extracts all the text from the document and returns it as a string. The reading order is taken from the order the text is drawn in the source file, so may not be accurate.

Parameters

<i>separator</i>	The character(s) used to separate the text lines obtained from the document. If this is <code>null</code> , <code>Environment.NewLine</code> is used as a default separator.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included. Otherwise, only the page contents are included.

Returns

A string containing all the text in the document. Characters are converted from the UTF-8 representation used in the document to equivalent UTF-16 strings.

Definition at line 1469 of file [MuPDFDocument.cs](#).

7.17.3.6 ExtractText() [2/2]

```
string MuPDFCore.MuPDFDocument.ExtractText (
    TesseractLanguage ocrLanguage,
    string separator = null,
    bool includeAnnotations = true )
```

Extracts all the text from the document and returns it as a string, using optical character recognition (OCR) to determine what text is written on the image.

Parameters

<i>separator</i>	The character(s) used to separate the text lines obtained from the document. If this is <code>null</code> , <code>Environment.NewLine</code> is used as a default separator.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is <code>null</code> , no OCR is performed.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included. Otherwise, only the page contents are included.

Returns

A string containing all the text in the document. Characters are converted from the UTF-8 representation used in the document to equivalent UTF-16 strings.

Definition at line 1518 of file [MuPDFDocument.cs](#).

7.17.3.7 ExtractTextAsync()

```
async Task< string > MuPDFCore.MuPDFDocument.ExtractTextAsync (
    TesseractLanguage ocrLanguage,
    string separator = null,
    bool includeAnnotations = true,
    CancellationToken cancellationToken = default,
    IProgress< OCRProgressInfo > progress = null )
```

Extracts all the text from the document and returns it as a string, using optical character recognition (OCR) to determine what text is written on the image. The OCR step is run asynchronously, e.g. to avoid blocking the UI thread.

Parameters

<i>separator</i>	The character(s) used to separate the text lines obtained from the document. If this is null, Environment.NewLine is used as a default separator.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>includeAnnotations</i>	If this is true, annotations (e.g. signatures) are included. Otherwise, only the page contents are included.
<i>cancellationToken</i>	A CancellationToken used to cancel the operation.
<i>progress</i>	An IProgress<OCRProgressInfo> used to report progress.

Returns

A string containing all the text in the document. Characters are converted from the UTF-8 representation used in the document to equivalent UTF-16 strings.

Definition at line 1569 of file [MuPDFDocument.cs](#).

7.17.3.8 GetMultiThreadedRenderer()

```
MuPDFMultiThreadedPageRenderer MuPDFCore.MuPDFDocument.GetMultiThreadedRenderer (
    int pageNumber,
    int threadCount,
    bool includeAnnotations = true )
```

Create a new [MuPDFMultiThreadedPageRenderer](#) that renders the specified page with the specified number of threads.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>threadCount</i>	The number of threads to use. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.

Returns

A [MuPDFMultiThreadedPageRenderer](#) that can be used to render the specified page with the specified number of threads.

Parameters

<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.
---------------------------	--

Definition at line 947 of file [MuPDFDocument.cs](#).

7.17.3.9 GetRenderedSize() [1/2]

```
int MuPDFCore.MuPDFDocument.GetRenderedSize (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat )
```

Determine how many bytes will be necessary to render the specified page at the specified zoom level, using the the specified pixel format.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixels data.

Returns

An integer representing the number of bytes that will be necessary to store the pixel data of the rendered image.

Definition at line 969 of file [MuPDFDocument.cs](#).

7.17.3.10 GetRenderedSize() [2/2]

```
static int MuPDFCore.MuPDFDocument.GetRenderedSize (
    Rectangle region,
    double zoom,
    PixelFormats pixelFormat ) [static]
```

Determine how many bytes will be necessary to render the specified region in page units at the specified zoom level, using the the specified pixel format.

Parameters

<i>region</i>	The region that will be rendered.
<i>zoom</i>	The scale at which the region will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixels data.

Returns

An integer representing the number of bytes that will be necessary to store the pixel data of the rendered image.

Definition at line 986 of file [MuPDFDocument.cs](#).

7.17.3.11 GetStructuredTextPage() [1/2]

```
MuPDFStructuredTextPage MuPDFCore.MuPDFDocument.GetStructuredTextPage (
    int pageNumber,
    bool includeAnnotations = true,
    StructuredTextFlags flags = StructuredTextFlags.None )
```

Creates a new MuPDFStructuredTextPage from the specified page. This contains information about the text layout that can be used for highlighting and searching. The reading order is taken from the order the text is drawn in the source file, so may not be accurate.

Parameters

<i>pageNumber</i>	The number of the page (starting at 0)
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included. Otherwise, only the page contents are included.
<i>flags</i>	Flags for the structured text extraction process.

Returns

A MuPDFStructuredTextPage containing a structured text representation of the page.

Definition at line 1380 of file [MuPDFDocument.cs](#).

7.17.3.12 GetStructuredTextPage() [2/2]

```
MuPDFStructuredTextPage MuPDFCore.MuPDFDocument.GetStructuredTextPage (
    int pageNumber,
    TesseractLanguage ocrLanguage,
    bool includeAnnotations = true,
    StructuredTextFlags flags = StructuredTextFlags.None,
    CancellationToken cancellationToken = default,
    IProgress< OCRProgressInfo > progress = null )
```

Creates a new MuPDFStructuredTextPage from the specified page, using optical character recognition (OCR) to determine what text is written on the image. This contains information about the text layout that can be used for highlighting and searching.

Parameters

<i>pageNumber</i>	The number of the page (starting at 0)
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included. Otherwise, only the page contents are included.
<i>flags</i>	Flags for the structured text extraction process.
<i>cancellationToken</i>	A <code>CancellationToken</code> used to cancel the operation. Providing a value other than the default is not supported on Windows x86 and will throw a runtime exception.
<i>progress</i>	An <code>IProgress<OCRProgressInfo></code> used to report progress. Providing a value other than null is not supported on Windows x86 and will throw a runtime exception.

Returns

A `MuPDFStructuredTextPage` containing a structured text representation of the page.

Definition at line 1405 of file [MuPDFDocument.cs](#).

7.17.3.13 GetStructuredTextPageAsync()

```
async Task< MuPDFStructuredTextPage > MuPDFCore.MuPDFDocument.GetStructuredTextPageAsync (
    int pageNumber,
    TesseractLanguage ocrLanguage,
    bool includeAnnotations = true,
    StructuredTextFlags flags = StructuredTextFlags.None,
    CancellationToken cancellationToken = default,
    IProgress< OCRProgressInfo > progress = null )
```

Creates a new `MuPDFStructuredTextPage` from the specified page, using optical character recognition (OCR) to determine what text is written on the image. This contains information about the text layout that can be used for highlighting and searching. The OCR step is run asynchronously, e.g. to avoid blocking the UI thread.

Parameters

<i>pageNumber</i>	The number of the page (starting at 0)
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included. Otherwise, only the page contents are included.
<i>flags</i>	Flags for the structured text extraction process.
<i>cancellationToken</i>	A <code>CancellationToken</code> used to cancel the operation. Providing a value other than the default is not supported on Windows x86 and will throw a runtime exception.
<i>progress</i>	An <code>IProgress<OCRProgressInfo></code> used to report progress. Providing a value other than null is not supported on Windows x86 and will throw a runtime exception.

Returns

A `MuPDFStructuredTextPage` containing a structured text representation of the page.

Definition at line 1439 of file [MuPDFDocument.cs](#).

7.17.3.14 Layout()

```
void MuPDFCore.MuPDFDocument.Layout (
    float width,
    float height,
    float em )
```

Sets the document layout for reflowable document types (e.g., HTML, MOBI). Does not have any effect for documents with a fixed layout (e.g., PDF).

Parameters

<i>width</i>	The width of each page, in points. Must be > 0.
<i>height</i>	The height of each page, in points. Must be > 0.
<i>em</i>	The default font size, in points.

Definition at line 713 of file [MuPDFDocument.cs](#).

7.17.3.15 LayoutSinglePage()

```
void MuPDFCore.MuPDFDocument.LayoutSinglePage (
    float width,
    float em )
```

Sets the document layout for reflowable document types (e.g., HTML, MOBI), so that the document is rendered to a single page, as tall as necessary. Does not have any effect for documents with a fixed layout (e.g., PDF).

Parameters

<i>width</i>	The width of the page, in points. Must be > 0.
<i>em</i>	The default font size, in points.

Definition at line 741 of file [MuPDFDocument.cs](#).

7.17.3.16 Render() [1/6]

```
byte[] MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat,
    bool includeAnnotations = true )
```

Render a page to an array of bytes.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Returns

A byte array containing the raw values for the pixels of the rendered image.

Definition at line 801 of file [MuPDFDocument.cs](#).

7.17.3.17 Render() [2/6]

```
void MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat,
    IntPtr destination,
    bool includeAnnotations = true )
```

Render a page to the specified destination.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>destination</i>	The address of the buffer where the pixel data will be written. There must be enough space available to write the values for all the pixels, otherwise this will fail catastrophically!
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 880 of file [MuPDFDocument.cs](#).

7.17.3.18 Render() [3/6]

```
Span< byte > MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat,
```

```
out IDisposable disposable,  
bool includeAnnotations = true )
```

Render a page to a Span<byte>.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>disposable</i>	An <i>IDisposable</i> that can be used to free the memory where the image is stored. You should keep track of this and dispose it when you have finished working with the image.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 928 of file [MuPDFDocument.cs](#).

7.17.3.19 Render() [4/6]

```
byte[] MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    Rectangle region,
    double zoom,
    PixelFormats pixelFormat,
    bool includeAnnotations = true )
```

Render (part of) a page to an array of bytes.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Returns

A byte array containing the raw values for the pixels of the rendered image.

Definition at line 767 of file [MuPDFDocument.cs](#).

7.17.3.20 Render() [5/6]

```
void MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    Rectangle region,
```

```
        double zoom,
        PixelFormats pixelFormat,
        IntPtr destination,
        bool includeAnnotations = true )
```

Render (part of) a page to the specified destination.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>destination</i>	The address of the buffer where the pixel data will be written. There must be enough space available to write the values for all the pixels, otherwise this will fail catastrophically!
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 821 of file [MuPDFDocument.cs](#).

7.17.3.21 Render() [6/6]

```
Span< byte > MuPDFCore.MuPDFDocument.Render (
    int pageNumber,
    Rectangle region,
    double zoom,
    PixelFormats pixelFormat,
    out IDisposable disposable,
    bool includeAnnotations = true )
```

Render (part of) a page to a `Span<byte>`.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>disposable</i>	An <code>IDisposable</code> that can be used to free the memory where the image is stored. You should keep track of this and dispose it when you have finished working with the image.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 900 of file [MuPDFDocument.cs](#).

7.17.3.22 SaveImage() [1/2]

```
void MuPDFCore.MuPDFDocument.SaveImage (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat,
    string fileName,
    RasterOutputFileTypes fileType,
    bool includeAnnotations = true )
```

Save a page to an image file in the specified format.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>fileName</i>	The path to the output file.
<i>fileType</i>	The output format of the file.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1150 of file [MuPDFDocument.cs](#).

7.17.3.23 SaveImage() [2/2]

```
void MuPDFCore.MuPDFDocument.SaveImage (
    int pageNumber,
    Rectangle region,
    double zoom,
    PixelFormats pixelFormat,
    string fileName,
    RasterOutputFileTypes fileType,
    bool includeAnnotations = true )
```

Save (part of) a page to an image file in the specified format.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>fileName</i>	The path to the output file.
<i>fileType</i>	The output format of the file.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1022 of file [MuPDFDocument.cs](#).

7.17.3.24 SaveImageAsJPEG() [1/2]

```
void MuPDFCore.MuPDFDocument.SaveImageAsJPEG (
    int pageNumber,
    double zoom,
    string fileName,
    int quality,
    bool includeAnnotations = true )
```

Save a page to an image file in JPEG format, with the specified quality.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>fileName</i>	The path to the output file.
<i>quality</i>	The quality of the JPEG output file (ranging from 0 to 100).
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1169 of file [MuPDFDocument.cs](#).

7.17.3.25 SaveImageAsJPEG() [2/2]

```
void MuPDFCore.MuPDFDocument.SaveImageAsJPEG (
    int pageNumber,
    Rectangle region,
    double zoom,
    string fileName,
    int quality,
    bool includeAnnotations = true )
```

Save (part of) a page to an image file in JPEG format, with the specified quality.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>fileName</i>	The path to the output file.
<i>quality</i>	The quality of the JPEG output file (ranging from 0 to 100).
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1091 of file [MuPDFDocument.cs](#).

7.17.3.26 TryUnlock() [1/2]

```
bool MuPDFCore.MuPDFDocument.TryUnlock (
    string password )
```

Attempts to unlock the document with the supplied password.

Parameters

<i>password</i>	The user or owner password to use to unlock the document.
-----------------	---

Returns

`true` if the document was successfully unlocked (or if it was never locked to begin with), `false` if the password was incorrect and the document is still locked.

This method can be used both to unlock an encrypted document and to check whether the supplied owner password is correct.

Definition at line 1617 of file [MuPDFDocument.cs](#).

7.17.3.27 TryUnlock() [2/2]

```
bool MuPDFCore.MuPDFDocument.TryUnlock (
    string password,
    out PasswordTypes passwordType )
```

Attempts to unlock the document with the supplied password.

Parameters

<i>password</i>	The user or owner password to use to unlock the document.
<i>passwordType</i>	If the method returns <code>true</code> , this can be used to determine whether the supplied password was the user password or the owner password. If the method returns <code>false</code> , this can be used to determine whether a user password and/or an owner password are required.

Returns

`true` if the document was successfully unlocked (or if it was never locked to begin with), `false` if the password was incorrect and the document is still locked.

This method can be used both to unlock an encrypted document and to check whether the supplied owner password is correct.

Definition at line 1630 of file [MuPDFDocument.cs](#).

7.17.3.28 WriteImage() [1/2]

```
void MuPDFCore.MuPDFDocument.WriteImage (
    int pageNumber,
    double zoom,
    PixelFormats pixelFormat,
    Stream outputStream,
    RasterOutputFileTypes fileType,
    bool includeAnnotations = true )
```

Write a page to an image stream in the specified format.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>outputStream</i>	The stream to which the image data will be written.
<i>fileType</i>	The output format of the image.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1343 of file [MuPDFDocument.cs](#).

7.17.3.29 WriteImage() [2/2]

```
void MuPDFCore.MuPDFDocument.WriteImage (
    int pageNumber,
    Rectangle region,
    double zoom,
    PixelFormats pixelFormat,
    Stream outputStream,
    RasterOutputFileTypes fileType,
    bool includeAnnotations = true )
```

Write (part of) a page to an image stream in the specified format.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>pixelFormat</i>	The format of the pixel data.
<i>outputStream</i>	The stream to which the image data will be written.
<i>fileType</i>	The output format of the image.
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1190 of file [MuPDFDocument.cs](#).

7.17.3.30 WriteImageAsJPEG() [1/2]

```
void MuPDFCore.MuPDFDocument.WriteImageAsJPEG (
    int pageNumber,
    double zoom,
    Stream outputStream,
    int quality,
    bool includeAnnotations = true )
```

Write a page to an image stream in JPEG format, with the specified quality.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>outputStream</i>	The stream to which the image data will be written.
<i>quality</i>	The quality of the JPEG output (ranging from 0 to 100).
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1362 of file [MuPDFDocument.cs](#).

7.17.3.31 WriteImageAsJPEG() [2/2]

```
void MuPDFCore.MuPDFDocument.WriteImageAsJPEG (
    int pageNumber,
    Rectangle region,
    double zoom,
    Stream outputStream,
    int quality,
    bool includeAnnotations = true )
```

Write (part of) a page to an image stream in JPEG format, with the specified quality.

Parameters

<i>pageNumber</i>	The number of the page to render (starting at 0).
<i>region</i>	The region of the page to render in page units.
<i>zoom</i>	The scale at which the page will be rendered. This will determine the size in pixel of the image.
<i>outputStream</i>	The stream to which the image data will be written.
<i>quality</i>	The quality of the JPEG output (ranging from 0 to 100).
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 1272 of file [MuPDFDocument.cs](#).

7.17.4 Property Documentation

7.17.4.1 ClipToBounds

```
bool MuPDFCore.MuPDFDocument.ClipToBounds = true [get], [set]
```

Defines whether the images resulting from rendering operations should be clipped to the page boundaries.

Definition at line 123 of file [MuPDFDocument.cs](#).

7.17.4.2 EncryptionState

```
EncryptionState MuPDFCore.MuPDFDocument.EncryptionState [get]
```

Describes the encryption state of the document.

Definition at line 128 of file [MuPDFDocument.cs](#).

7.17.4.3 OptionalContentGroupData

```
MuPDFOptionalContentGroupData MuPDFCore.MuPDFDocument.OptionalContentGroupData [get]
```

Contains information about optional content groups (aka layers). If this document does not contain any optional content groups, this object will be null. The optional content group data is loaded from the document at the first access.

Definition at line 168 of file [MuPDFDocument.cs](#).

7.17.4.4 Outline

```
MuPDFOutline MuPDFCore.MuPDFDocument.Outline [get]
```

The document outline (table of contents). If this document does not have an outline, this object will be empty, but not null. The outline is loaded from the document at the first access.

Definition at line 147 of file [MuPDFDocument.cs](#).

7.17.4.5 Pages

```
MuPDFPageCollection MuPDFCore.MuPDFDocument.Pages [get]
```

The pages contained in the document.

Definition at line 118 of file [MuPDFDocument.cs](#).

7.17.4.6 Restrictions

```
DocumentRestrictions MuPDFCore.MuPDFDocument.Restrictions [get]
```

Describes the operations that are restricted on the document. This is not actually enforced by the library, but library users should only allow these operations if the document has been unlocked with the owner password (i.e. if [RestrictionState](#) is [RestrictionState.Unlocked](#)).

Definition at line 140 of file [MuPDFDocument.cs](#).

7.17.4.7 RestrictionState

```
RestrictionState MuPDFCore.MuPDFDocument.RestrictionState [get]
```

Describes the restriction state of the document.

Definition at line 133 of file [MuPDFDocument.cs](#).

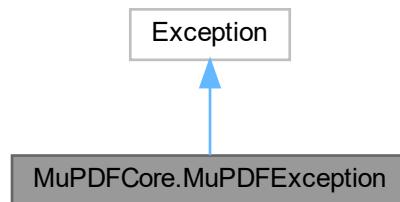
The documentation for this class was generated from the following files:

- [MuPDFCore/MuPDFDocument.Create.cs](#)
- [MuPDFCore/MuPDFDocument.cs](#)

7.18 MuPDFCore.MuPDFException Class Reference

The exception that is thrown when a [MuPDF](#) operation fails.

Inheritance diagram for MuPDFCore.MuPDFException:



Public Attributes

- readonly [ExitCodes ErrorCode](#)
The [ExitCodes](#) returned by the native function.

7.18.1 Detailed Description

The exception that is thrown when a [MuPDF](#) operation fails.

Definition at line 595 of file [MuPDF.cs](#).

7.18.2 Member Data Documentation

7.18.2.1 ErrorCode

```
readonly ExitCodes MuPDFCore.MuPDFException.ErrorCode
```

The [ExitCodes](#) returned by the native function.

Definition at line 600 of file [MuPDF.cs](#).

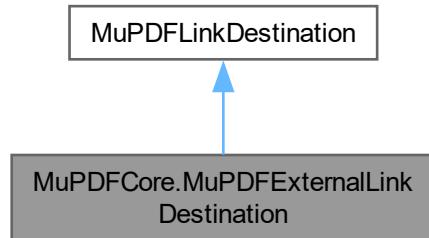
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDF.cs](#)

7.19 MuPDFCore.MuPDFExternalLinkDestination Class Reference

A link destination to an external resource (e.g., a website).

Inheritance diagram for MuPDFCore.MuPDFExternalLinkDestination:



Properties

- string [Uri](#) [get]
The Uri of the external resource.
- override [DestinationType](#) [Type](#) [get]

Additional Inherited Members

7.19.1 Detailed Description

A link destination to an external resource (e.g., a website).

Definition at line [231](#) of file [MuPDFLinks.cs](#).

7.19.2 Property Documentation

7.19.2.1 Type

```
override DestinationType MuPDFCore.MuPDFExternalLinkDestination.Type [get]
```

Definition at line [239](#) of file [MuPDFLinks.cs](#).

7.19.2.2 Uri

```
string MuPDFCore.MuPDFExternalLinkDestination.Uri [get]
```

The Uri of the external resource.

Definition at line [236](#) of file [MuPDFLinks.cs](#).

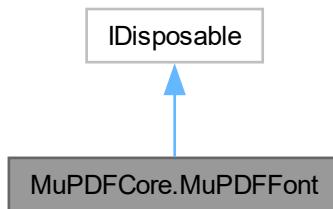
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFLinks.cs

7.20 MuPDFCore.MuPDFFont Class Reference

Represents a font.

Inheritance diagram for MuPDFCore.MuPDFFont:



Public Member Functions

- IntPtr [GetFreeTypeHandle \(\)](#)
Get a pointer to the FreeType FT_Face object for this font. You will need native bindings to the FreeType library to use this.
- IntPtr [GetType3Handle \(\)](#)
Get a pointer to the Type3 procs for the font. You will need some more specialised MuPDF bindings to do anything with it.
- void [Dispose \(\)](#)

Properties

- bool [IsBold \[get\]](#)
Returns whether the font is bold or not.
- bool [IsItalic \[get\]](#)
Returns whether the font is italic or not.
- bool [IsSerif \[get\]](#)
Returns whether the font is serif or not.
- bool [IsMonospaced \[get\]](#)
Returns whether the font is monospaced or not.
- string [Name \[get\]](#)
The name of the font.

7.20.1 Detailed Description

Represents a font.

Definition at line 28 of file [MuPDFFont.cs](#).

7.20.2 Member Function Documentation

7.20.2.1 Dispose()

```
void MuPDFCore.MuPDFFont.Dispose ( )
```

Definition at line 205 of file [MuPDFFont.cs](#).

7.20.2.2 GetFreeTypeHandle()

```
IntPtr MuPDFCore.MuPDFFont.GetFreeTypeHandle ( )
```

Get a pointer to the FreeType FT_Face object for this font. You will need native bindings to the FreeType library to use this.

Returns

A pointer to the FreeType FT_Face object for this font, or IntPtr.Zero if this is a Type3 font.

Exceptions

MuPDFException	Thrown if an error occurs while accessing the FreeType handle for the font.
--------------------------------	---

Definition at line 112 of file [MuPDFFont.cs](#).

7.20.2.3 GetType3Handle()

```
IntPtr MuPDFCore.MuPDFFont.GetType3Handle ( )
```

Get a pointer to the Type3 procs for the font. You will need some more specialised [MuPDF](#) bindings to do anything with it.

Returns

A pointer to the Type3 procs for the font, or IntPtr.Zero if this is not a Type3 font.

Exceptions

MuPDFException	Thrown if an error occurs while accessing the Type3 font procs.
--------------------------------	---

Definition at line 141 of file [MuPDFFont.cs](#).

7.20.3 Property Documentation

7.20.3.1 IsBold

```
bool MuPDFCore.MuPDFFont.IsBold [get]
```

Returns whether the font is bold or not.

Definition at line 35 of file [MuPDFFont.cs](#).

7.20.3.2 IsItalic

```
bool MuPDFCore.MuPDFFont.IsItalic [get]
```

Returns whether the font is italic or not.

Definition at line 40 of file [MuPDFFont.cs](#).

7.20.3.3 IsMonospaced

```
bool MuPDFCore.MuPDFFont.IsMonospaced [get]
```

Returns whether the font is monospaced or not.

Definition at line 50 of file [MuPDFFont.cs](#).

7.20.3.4 IsSerif

```
bool MuPDFCore.MuPDFFont.IsSerif [get]
```

Returns whether the font is serif or not.

Definition at line 45 of file [MuPDFFont.cs](#).

7.20.3.5 Name

```
string MuPDFCore.MuPDFFont.Name [get]
```

The name of the font.

Definition at line 55 of file [MuPDFFont.cs](#).

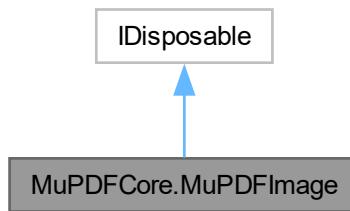
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFFont.cs](#)

7.21 MuPDFCore.MuPDFImage Class Reference

Represents an image embedded within a document.

Inheritance diagram for MuPDFCore.MuPDFImage:



Public Types

- enum [ImageOrientation](#)
Describes the orientation of the image (as encoded within the image file).

Public Member Functions

- void [Save](#) (string fileName, [RasterOutputFileTypes](#) fileType, bool? convertToRGB=null)
Save the image to a file.
- void [SaveAsJPEG](#) (string fileName, int quality, bool? convertToRGB=null)
Save the image to a JPEG file.
- void [Write](#) (Stream outputStream, [RasterOutputFileTypes](#) fileType, bool? convertToRGB=null)
Write the image to a Stream.
- void [WriteAsJPEG](#) (Stream outputStream, int quality, bool? convertToRGB=null)
Write the image to a Stream in JPEG format.
- unsafe byte[] [GetBytes](#) ([PixelFormats](#) pixelFormat)
Get a byte representation of the image pixels.
- unsafe byte[] [GetBytes](#) ()
Get a byte representation of the image pixels.
- void [Dispose](#) ()

Properties

- int [Width](#) [get]
Width of the image in pixels.
- int [Height](#) [get]
Height of the image in pixels.
- int [XRes](#) [get]
Horizontal resolution of the image.
- int [YRes](#) [get]
Vertical resolution of the image.
- [ImageOrientation Orientation](#) [get]
Orientation of the image (as encoded within the image file).
- [MuPDFColorSpace ColorSpace](#) [get]
The colour space in which the image is defined.
- MuPDFImageStructuredTextBlock [ParentBlock](#) [get]
The MuPDFImageStructuredTextBlock from which this image was obtained.

7.21.1 Detailed Description

Represents an image embedded within a document.

Definition at line 28 of file [MuPDFImage.cs](#).

7.21.2 Member Enumeration Documentation

7.21.2.1 [ImageOrientation](#)

```
enum MuPDFCore.MuPDFImage.ImageOrientation
```

Describes the orientation of the image (as encoded within the image file).

Definition at line 35 of file [MuPDFImage.cs](#).

7.21.3 Member Function Documentation

7.21.3.1 [Dispose\(\)](#)

```
void MuPDFCore.MuPDFImage.Dispose ( )
```

Definition at line 516 of file [MuPDFImage.cs](#).

7.21.3.2 [GetBytes\(\)](#) [1/2]

```
unsafe byte[ ] MuPDFCore.MuPDFImage.GetBytes ( )
```

Get a byte representation of the image pixels.

Returns

A byte representation of the image pixels, in the native image colour space.

Exceptions

<i>MuPDFException</i>	Thrown if an error occurs while rendering the image.
-----------------------	--

Definition at line 457 of file [MuPDFImage.cs](#).

7.21.3.3 GetBytes() [2/2]

```
unsafe byte[ ] MuPDFCore.MuPDFImage.GetBytes (
    PixelFormats pixelFormat )
```

Get a byte representation of the image pixels.

Returns

A byte representation of the image pixels in the specified pixel format.

Exceptions

<i>MuPDFException</i>	Thrown if an error occurs while rendering the image.
-----------------------	--

Definition at line 397 of file [MuPDFImage.cs](#).

7.21.3.4 Save()

```
void MuPDFCore.MuPDFImage.Save (
    string fileName,
    RasterOutputFileTypes fileType,
    bool? convertToRGB = null )
```

Save the image to a file.

Parameters

<i>fileName</i>	The name of the output file.
<i>fileType</i>	The output file format.
<i>convertToRGB</i>	If this is <code>true</code> , the image is converted to the RGB colour space before being saved. If this is <code>false</code> , the image is saved in the same colour space as it was encoded in the document. If this is <code>null</code> (the default), the image is converted to RGB only if the target colour space does not support the colour space of the image.

Exceptions

<i>MuPDFException</i>	Thrown if an error occurs while rendering the image or saving it.
-----------------------	---

Exceptions

<i>ArgumentException</i>	Thrown if attempting to export an image in a format that does not support the colour space of the image.
--------------------------	--

Definition at line 167 of file [MuPDFImage.cs](#).

7.21.3.5 SaveAsJPEG()

```
void MuPDFCore.MuPDFImage.SaveAsJPEG (
    string fileName,
    int quality,
    bool? convertToRGB = null )
```

Save the image to a JPEG file.

Parameters

<i>fileName</i>	The name of the output file.
<i>quality</i>	The quality of the JPEG image (from 0 to 100).
<i>convertToRGB</i>	If this is true, the image is converted to the RGB colour space before being saved. If this is false, the image is saved in the same colour space as it was encoded in the document. If this is null (the default), the image is converted to RGB only if the target colour space does not support the colour space of the image.

Exceptions

<i>ArgumentOutOfRangeException</i>	Thrown if the quality value is < 0 or > 100.
<i>MuPDFException</i>	Thrown if an error occurs while rendering the image or saving it.

Definition at line 220 of file [MuPDFImage.cs](#).

7.21.3.6 Write()

```
void MuPDFCore.MuPDFImage.Write (
    Stream outputStream,
    RasterOutputFileTypes fileType,
    bool? convertToRGB = null )
```

Write the image to a Stream.

Parameters

<i>outputStream</i>	The output Stream.
<i>fileType</i>	The image format.
<i>convertToRGB</i>	If this is true, the image is converted to the RGB colour space before being saved. If this is false, the image is saved in the same colour space as it was encoded in the document. If this is null (the default), the image is converted to RGB only if the target colour space does not support the colour space of the image.
Generated by Doxygen	

Exceptions

MuPDFException	Thrown if an error occurs while rendering the image or saving it.
--------------------------------	---

Definition at line 266 of file [MuPDFImage.cs](#).

7.21.3.7 WriteAsJPEG()

```
void MuPDFCore.MuPDFImage.WriteAsJPEG (
    Stream outputStream,
    int quality,
    bool? convertToRGB = null )
```

Write the image to a Stream in JPEG format.

Parameters

<i>outputStream</i>	The output Stream.
<i>quality</i>	The quality of the JPEG image (from 0 to 100).
<i>convertToRGB</i>	If this is <code>true</code> , the image is converted to the RGB colour space before being saved. If this is <code>false</code> , the image is saved in the same colour space as it was encoded in the document. If this is <code>null</code> (the default), the image is converted to RGB only if the target colour space does not support the colour space of the image.

Exceptions

ArgumentOutOfRangeException	Thrown if the quality value is < 0 or > 100
MuPDFException	Thrown if an error occurs while rendering the image or saving it.

Definition at line 336 of file [MuPDFImage.cs](#).

7.21.4 Property Documentation

7.21.4.1 ColorSpace

[MuPDFColorSpace](#) MuPDFCore.MuPDFImage.ColorSpace [get]

The colour space in which the image is defined.

Definition at line 111 of file [MuPDFImage.cs](#).

7.21.4.2 Height

```
int MuPDFCore.MuPDFImage.Height [get]
```

Height of the image in pixels.

Definition at line 91 of file [MuPDFImage.cs](#).

7.21.4.3 Orientation

```
ImageOrientation MuPDFCore.MuPDFImage.Orientation [get]
```

Orientation of the image (as encoded within the image file).

Definition at line 106 of file [MuPDFImage.cs](#).

7.21.4.4 ParentBlock

```
MuPDFImageStructuredTextBlock MuPDFCore.MuPDFImage.ParentBlock [get]
```

The MuPDFImageStructuredTextBlock from which this image was obtained.

Definition at line 116 of file [MuPDFImage.cs](#).

7.21.4.5 Width

```
int MuPDFCore.MuPDFImage.Width [get]
```

Width of the image in pixels.

Definition at line 86 of file [MuPDFImage.cs](#).

7.21.4.6 XRes

```
int MuPDFCore.MuPDFImage.XRes [get]
```

Horizontal resolution of the image.

Definition at line 96 of file [MuPDFImage.cs](#).

7.21.4.7 YRes

```
int MuPDFCore.MuPDFImage.YRes [get]
```

Vertical resolution of the image.

Definition at line 101 of file [MuPDFImage.cs](#).

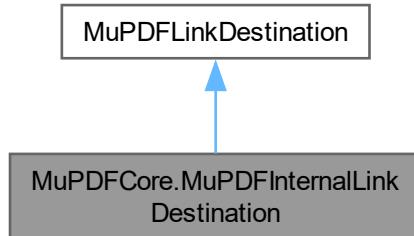
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFImage.cs](#)

7.22 MuPDFCore.MuPDFInternalLinkDestination Class Reference

An internal link destination.

Inheritance diagram for MuPDFCore.MuPDFInternalLinkDestination:



Public Types

- enum [InternalDestinationType](#)

Defines internal link destination types.

Properties

- override [DestinationType Type](#) [get]
- int [Chapter](#) [get]

Locations within the document are referred to in terms of chapter and page, rather than just a page number. For some documents (such as epub documents with large numbers of pages broken into many chapters) this can make navigation much faster as only the required chapter needs to be decoded at a time.

- int [Page](#) [get]

The page number of an internal link, relative to the specified [Chapter](#), or -1 for external links or links with no destination.

- int [PageNumber](#) [get]

The overall page number of an internal link. This is determined on first access, and it might cause a large number of chapters to be laid out to determine it.

- float [X](#) [get]
X coordinate of the link target on the specified page.
- float [Y](#) [get]
Y coordinate of the link target on the specified page.
- float [Width](#) [get]
Width of the link target on the specified page.
- float [Height](#) [get]
Height of the link target on the specified page.
- float [Zoom](#) [get]
Target magnification factor.
- [InternalDestinationType InternalType](#) [get]
The type of internal link destination.

7.22.1 Detailed Description

An internal link destination.

Definition at line 270 of file [MuPDFLinks.cs](#).

7.22.2 Member Enumeration Documentation

7.22.2.1 InternalDestinationType

```
enum MuPDFCore.MuPDFInternalLinkDestination.InternalDestinationType
```

Defines internal link destination types.

Definition at line 275 of file [MuPDFLinks.cs](#).

7.22.3 Property Documentation

7.22.3.1 Chapter

```
int MuPDFCore.MuPDFInternalLinkDestination.Chapter [get]
```

Locations within the document are referred to in terms of chapter and page, rather than just a page number. For some documents (such as epub documents with large numbers of pages broken into many chapters) this can make navigation much faster as only the required chapter needs to be decoded at a time.

Definition at line 324 of file [MuPDFLinks.cs](#).

7.22.3.2 Height

```
float MuPDFCore.MuPDFInternalLinkDestination.Height [get]
```

Height of the link target on the specified page.

Definition at line 367 of file [MuPDFLinks.cs](#).

7.22.3.3 InternalType

```
InternalDestinationType MuPDFCore.MuPDFInternalLinkDestination.InternalType [get]
```

The type of internal link destination.

Definition at line 377 of file [MuPDFLinks.cs](#).

7.22.3.4 Page

```
int MuPDFCore.MuPDFInternalLinkDestination.Page [get]
```

The page number of an internal link, relative to the specified [Chapter](#), or -1 for external links or links with no destination.

Definition at line 329 of file [MuPDFLinks.cs](#).

7.22.3.5 PageNumber

```
int MuPDFCore.MuPDFInternalLinkDestination.PageNumber [get]
```

The overall page number of an internal link. This is determined on first access, and it might cause a large number of chapters to be laid out to determine it.

Definition at line 336 of file [MuPDFLinks.cs](#).

7.22.3.6 Type

```
override DestinationType MuPDFCore.MuPDFInternalLinkDestination.Type [get]
```

Definition at line 319 of file [MuPDFLinks.cs](#).

7.22.3.7 Width

```
float MuPDFCore.MuPDFInternalLinkDestination.Width [get]
```

Width of the link target on the specified page.

Definition at line 362 of file [MuPDFLinks.cs](#).

7.22.3.8 X

```
float MuPDFCore.MuPDFInternalLinkDestination.X [get]
```

X coordinate of the link target on the specified page.

Definition at line 352 of file [MuPDFLinks.cs](#).

7.22.3.9 Y

```
float MuPDFCore.MuPDFInternalLinkDestination.Y [get]
```

Y coordinate of the link target on the specified page.

Definition at line 357 of file [MuPDFLinks.cs](#).

7.22.3.10 Zoom

```
float MuPDFCore.MuPDFInternalLinkDestination.Zoom [get]
```

Target magnification factor.

Definition at line 372 of file [MuPDFLinks.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFLinks.cs

7.23 MuPDFCore.MuPDFLink Class Reference

Represents a link within a [MuPDF](#) document.

Properties

- `Rectangle ActiveArea [get]`
The area on which users can click to activate the link.
- `MuPDFLinkDestination Destination [get]`
The link destination.
- `bool IsVisible [get]`
Whether the link is visible or hidden (e.g., because it is part of a hidden optional content group).

7.23.1 Detailed Description

Represents a link within a [MuPDF](#) document.

Definition at line 119 of file [MuPDFLinks.cs](#).

7.23.2 Property Documentation

7.23.2.1 ActiveArea

`Rectangle MuPDFCore.MuPDFLink.ActiveArea [get]`

The area on which users can click to activate the link.

Definition at line 124 of file [MuPDFLinks.cs](#).

7.23.2.2 Destination

`MuPDFLinkDestination MuPDFCore.MuPDFLink.Destination [get]`

The link destination.

Definition at line 129 of file [MuPDFLinks.cs](#).

7.23.2.3 IsVisible

`bool MuPDFCore.MuPDFLink.IsVisible [get]`

Whether the link is visible or hidden (e.g., because it is part of a hidden optional content group).

Definition at line 134 of file [MuPDFLinks.cs](#).

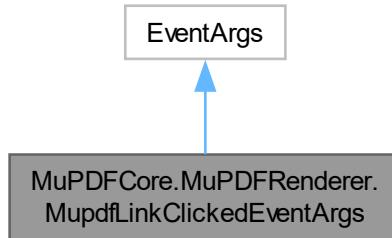
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFLinks.cs](#)

7.24 MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs Class Reference

EventArgs for the [PDFRenderer.LinkClicked](#) event.

Inheritance diagram for MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs:



Public Member Functions

- [MupdfLinkClickedEventArgs \(MuPDFLinkDestination linkDestination\)](#)
Create a new [MupdfLinkClickedEventArgs](#) instance.

Properties

- [MuPDFLinkDestination LinkDestination \[get\]](#)
The link destination.

7.24.1 Detailed Description

EventArgs for the [PDFRenderer.LinkClicked](#) event.

Definition at line 408 of file [PDFRenderer.Properties.cs](#).

7.24.2 Constructor & Destructor Documentation

7.24.2.1 MupdfLinkClickedEventArgs()

```
MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs.MupdfLinkClickedEventArgs (MuPDFLinkDestination linkDestination )
```

Create a new [MupdfLinkClickedEventArgs](#) instance.

Parameters

<i>linkDestination</i>	The link destination.
------------------------	-----------------------

Definition at line 419 of file [PDFRenderer.Properties.cs](#).

7.24.3 Property Documentation

7.24.3.1 LinkDestination

`MuPDFLinkDestination MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs.LinkDestination [get]`

The link destination.

Definition at line 413 of file [PDFRenderer.Properties.cs](#).

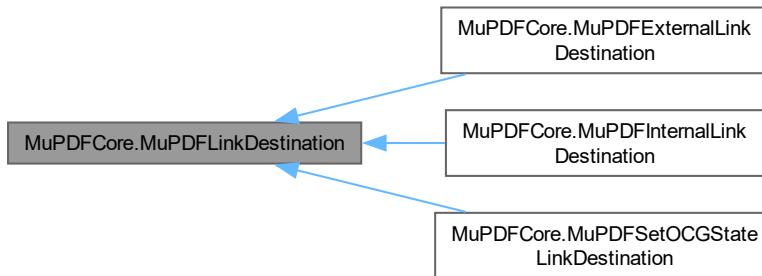
The documentation for this class was generated from the following file:

- [MuPDFCore.MuPDFRenderer/PDFRenderer.Properties.cs](#)

7.25 MuPDFCore.MuPDFLinkDestination Class Reference

Represents a generic link destination.

Inheritance diagram for MuPDFCore.MuPDFLinkDestination:



Public Types

- enum [DestinationType](#)

Defines the types of link destinations.

Properties

- abstract [DestinationType Type](#) [get]
The type of link destination.

7.25.1 Detailed Description

Represents a generic link destination.

Definition at line 192 of file [MuPDFLinks.cs](#).

7.25.2 Member Enumeration Documentation

7.25.2.1 DestinationType

```
enum MuPDFCore.MuPDFLinkDestination.DestinationType
```

Defines the types of link destinations.

Definition at line 197 of file [MuPDFLinks.cs](#).

7.25.3 Property Documentation

7.25.3.1 Type

```
abstract DestinationType MuPDFCore.MuPDFLinkDestination.Type [get]
```

The type of link destination.

Definition at line 218 of file [MuPDFLinks.cs](#).

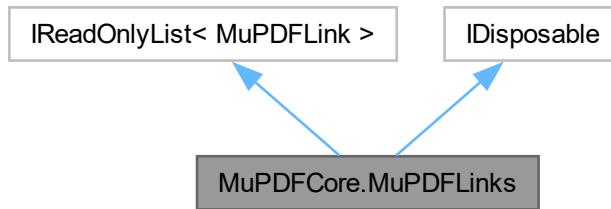
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFLinks.cs

7.26 MuPDFCore.MuPDFLinks Class Reference

A collection of [MuPDFLink](#) objects.

Inheritance diagram for MuPDFCore.MuPDFLinks:



Public Member Functions

- `IEnumerator< MuPDFLink > GetEnumerator ()`
- `void Dispose ()`

Properties

- `MuPDFLink this[int index] [get]`
- `int Count [get]`

7.26.1 Detailed Description

A collection of [MuPDFLink](#) objects.

Definition at line 28 of file [MuPDFLinks.cs](#).

7.26.2 Member Function Documentation

7.26.2.1 Dispose()

```
void MuPDFCore.MuPDFLinks.Dispose ( )
```

Definition at line 109 of file [MuPDFLinks.cs](#).

7.26.2.2 GetEnumerator()

```
IEnumerator< MuPDFLink > MuPDFCore.MuPDFLinks.GetEnumerator ( )
```

Definition at line 77 of file [MuPDFLinks.cs](#).

7.26.3 Property Documentation

7.26.3.1 Count

```
int MuPDFCore.MuPDFLinks.Count [get]
```

Definition at line 36 of file [MuPDFLinks.cs](#).

7.26.3.2 this[int index]

```
MuPDFLink MuPDFCore.MuPDFLinks.this[int index] [get]
```

Definition at line 33 of file [MuPDFLinks.cs](#).

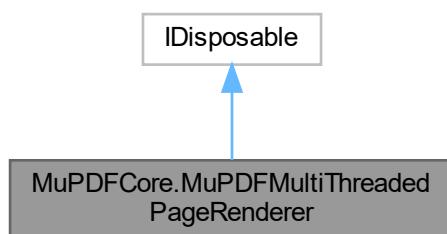
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFLinks.cs

7.27 MuPDFCore.MuPDFMultiThreadedPageRenderer Class Reference

A class that holds the necessary resources to render a page of a [MuPDF](#) document using multiple threads.

Inheritance diagram for MuPDFCore.MuPDFMultiThreadedPageRenderer:



Public Member Functions

- void [Render \(RoundedSize targetSize, Rectangle region, IntPtr\[\] destinations, PixelFormats pixelFormat\)](#)
Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this [MuPDFMultiThreadedPageRenderer](#), without marshaling. This method will not return until all the rendering threads have finished.
- delegate Span< byte > [GetSpanItem \(int index\)](#)
Gets an element from a collection of Span<byte>
- [GetSpanItem Render \(RoundedSize targetSize, Rectangle region, out IDisposable\[\] disposables, PixelFormats pixelFormat\)](#)
Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this [MuPDFMultiThreadedPageRenderer](#), without marshaling. This method will not return until all the rendering threads have finished. Since creating an array of Span<T> is not allowed, this method returns a delegate that accepts an integer parameter (representing the index of the span in the "array") and returns the Span<T> corresponding to that index.
- void [Abort \(\)](#)
Signal to the rendering threads that they should abort rendering as soon as possible.
- [RenderProgress GetProgress \(\)](#)
Get the current rendering progress of all the threads.
- void [Dispose \(\)](#)

Properties

- int [ThreadCount \[get\]](#)
The number of threads that are used to render the image.

7.27.1 Detailed Description

A class that holds the necessary resources to render a page of a [MuPDF](#) document using multiple threads.

Definition at line 276 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.2 Member Function Documentation

7.27.2.1 Abort()

```
void MuPDFCore.MuPDFMultiThreadedPageRenderer.Abort ( )
```

Signal to the rendering threads that they should abort rendering as soon as possible.

Definition at line 543 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.2.2 Dispose()

```
void MuPDFCore.MuPDFMultiThreadedPageRenderer.Dispose ( )
```

Definition at line 592 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.2.3 GetProgress()

```
RenderProgress MuPDFCore.MuPDFMultiThreadedPageRenderer.GetProgress ( )
```

Get the current rendering progress of all the threads.

Returns

A [RenderProgress](#) object containing the rendering progress of all the threads.

Definition at line 555 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.2.4 GetSpanItem()

```
delegate Span< byte > MuPDFCore.MuPDFMultiThreadedPageRenderer.GetSpanItem ( int index )
```

Gets an element from a collection of `Span<byte>`

Parameters

<code>index</code>	The index of the element to get.
--------------------	----------------------------------

Returns

An element from a collection of `Span<byte>`

7.27.2.5 Render() [1/2]

```
void MuPDFCore.MuPDFMultiThreadedPageRenderer.Render ( RoundedSize targetSize, Rectangle region, IntPtr[] destinations, PixelFormats pixelFormat )
```

Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this [MuPDFMultiThreadedPageRenderer](#), without marshaling. This method will not return until all the rendering threads have finished.

Parameters

<i>targetSize</i>	The total size of the image that should be rendered.
<i>region</i>	The region in page units that should be rendered.
<i>destinations</i>	An array containing the addresses of the buffers where the rendered tiles will be written. There must be enough space available in each buffer to write the values for all the pixels of the tile, otherwise this will fail catastrophically! As long as the <i>targetSize</i> is the same, the size in pixel of the tiles is guaranteed to also be the same.
<i>pixelFormat</i>	The format of the pixel data.

Definition at line 383 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.2.6 Render() [2/2]

```
GetSpanItem MuPDFCore.MuPDFMultiThreadedPageRenderer.Render (
    RoundedSize targetSize,
    Rectangle region,
    out IDisposable[] disposables,
    PixelFormats pixelFormat )
```

Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this [MuPDFMultiThreadedPageRenderer](#), without marshaling. This method will not return until all the rendering threads have finished. Since creating an array of Span<T> is not allowed, this method returns a delegate that accepts an integer parameter (representing the index of the span in the "array") and returns the Span<T> corresponding to that index.

Parameters

<i>targetSize</i>	The total size of the image that should be rendered.
<i>region</i>	The region in page units that should be rendered.
<i>disposables</i>	A collection of IDisposables that can be used to free the memory where the rendered tiles are stored. You should keep track of these and dispose them when you have finished working with the images.
<i>pixelFormat</i>	The format of the pixel data.

Returns

A delegate that accepts an integer parameter (representing the index of the span in the "array") and returns the Span<T> corresponding to that index.

Definition at line 509 of file [MuPDFMultiThreadedPageRenderer.cs](#).

7.27.3 Property Documentation

7.27.3.1 ThreadCount

```
int MuPDFCore.MuPDFMultiThreadedPageRenderer.ThreadCount [get]
```

The number of threads that are used to render the image.

Definition at line 306 of file [MuPDFMultiThreadedPageRenderer.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFMultiThreadedPageRenderer.cs](#)

7.28 MuPDFCore.MuPDFOptionalContentGroup Class Reference

Represents an optional content group (also known as layer).

Properties

- string [Name](#) [get]
The name of the optional content group.
- bool [IsEnabled](#) [get, set]
Gets or sets whether the optional content group is enabled.

7.28.1 Detailed Description

Represents an optional content group (also known as layer).

Definition at line 359 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.28.2 Property Documentation

7.28.2.1 IsEnabled

```
bool MuPDFCore.MuPDFOptionalContentGroup.IsEnabled [get], [set]
```

Gets or sets whether the optional content group is enabled.

Definition at line 372 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.28.2.2 Name

```
string MuPDFCore.MuPDFOptionalContentGroup.Name [get]
```

The name of the optional content group.

Definition at line 367 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

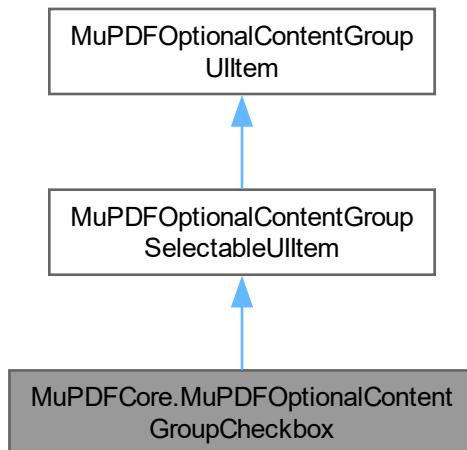
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.29 MuPDFCore.MuPDFOptionalContentGroupCheckbox Class Reference

An optional content group UI element that should be represented as a check box.

Inheritance diagram for MuPDFCore.MuPDFOptionalContentGroupCheckbox:



Additional Inherited Members

7.29.1 Detailed Description

An optional content group UI element that should be represented as a check box.

Definition at line 462 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.30 MuPDFCore.MuPDFOptionalContentGroupConfiguration Class Reference

Represents an optional content group configuration.

Public Member Functions

- void [Activate \(\)](#)
Activates this optional content group configuration.

Properties

- string [Name \[get\]](#)
The name of this optional content group configuration.
- string [Creator \[get\]](#)
The creator of this optional content group configuration.
- bool [IsDefault \[get\]](#)
Gets whether this is the default optional content group configuration for the document.
- [MuPDFOptionalContentGroupUIItem\[\] UI \[get\]](#)
Optional content group "UI" elements associated with this configuration.

7.30.1 Detailed Description

Represents an optional content group configuration.

Definition at line 111 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.30.2 Member Function Documentation

7.30.2.1 Activate()

```
void MuPDFCore.MuPDFOptionalContentGroupConfiguration.Activate ( )
```

Activates this optional content group configuration.

Definition at line 342 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.30.3 Property Documentation

7.30.3.1 Creator

```
string MuPDFCore.MuPDFOptionalContentGroupConfiguration.Creator [get]
```

The creator of this optional content group configuration.

Definition at line 121 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.30.3.2 IsDefault

```
bool MuPDFCore.MuPDFOptionalContentGroupConfiguration.IsDefault [get]
```

Gets whether this is the default optional content group configuration for the document.

Definition at line 126 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.30.3.3 Name

```
string MuPDFCore.MuPDFOptionalContentGroupConfiguration.Name [get]
```

The name of this optional content group configuration.

Definition at line 116 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.30.3.4 UI

```
MuPDFOptionalContentGroupUIItem [] MuPDFCore.MuPDFOptionalContentGroupConfiguration.UI [get]
```

Optional content group "UI" elements associated with this configuration.

Definition at line 131 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs

7.31 MuPDFCore.MuPDFOptionalContentGroupData Class Reference

Contains information about optional content groups in a PDF document (also known as layers).

Properties

- `MuPDFOptionalContentGroupConfiguration DefaultConfiguration [get]`
The default optional content group configuration.
- `MuPDFOptionalContentGroupConfiguration[] AlternativeConfigurations [get]`
Alternative optional content group configurations.
- `MuPDFOptionalContentGroup[] OptionalContentGroups [get]`
All optional content groups (layers) defined in this document.

7.31.1 Detailed Description

Contains information about optional content groups in a PDF document (also known as layers).

Definition at line 11 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.31.2 Property Documentation

7.31.2.1 AlternativeConfigurations

`MuPDFOptionalContentGroupConfiguration [] MuPDFCore.MuPDFOptionalContentGroupData.AlternativeConfigurations [get]`

Alternative optional content group configurations.

Definition at line 21 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.31.2.2 DefaultConfiguration

`MuPDFOptionalContentGroupConfiguration MuPDFCore.MuPDFOptionalContentGroupData.DefaultConfiguration [get]`

The default optional content group configuration.

Definition at line 16 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.31.2.3 OptionalContentGroups

`MuPDFOptionalContentGroup [] MuPDFCore.MuPDFOptionalContentGroupData.OptionalContentGroups [get]`

All optional content groups (layers) defined in this document.

Definition at line 26 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

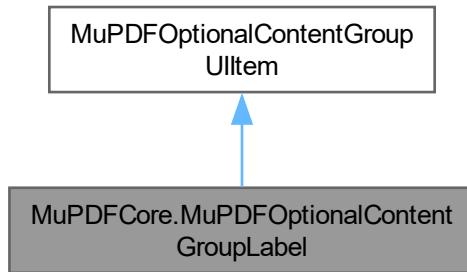
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.32 MuPDFCore.MuPDFOptionalContentGroupLabel Class Reference

An optional content group UI element that should be represented as a label.

Inheritance diagram for MuPDFCore.MuPDFOptionalContentGroupLabel:



Additional Inherited Members

7.32.1 Detailed Description

An optional content group UI element that should be represented as a label.

Definition at line 478 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

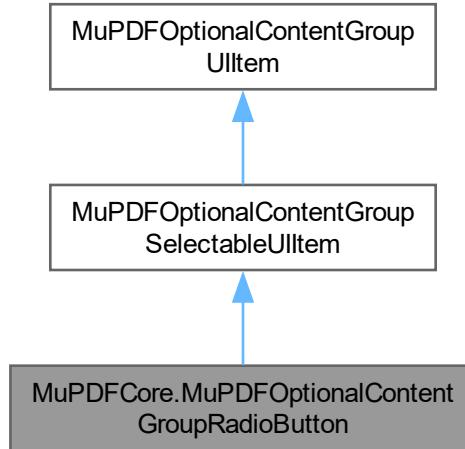
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs

7.33 MuPDFCore.MuPDFOptionalContentGroupRadioButton Class Reference

An optional content group UI element that should be represented as a radio button.

Inheritance diagram for MuPDFCore.MuPDFOptionalContentGroupRadioButton:



Additional Inherited Members

7.33.1 Detailed Description

An optional content group UI element that should be represented as a radio button.

Definition at line 470 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.34 MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem Class Reference

An optional content group UI element that can be enabled or disabled.

Inheritance diagram for MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem:



Public Member Functions

- virtual void [Toggle \(\)](#)
Toggle the state of the optional content group UI element.

Properties

- virtual bool [IsEnabled \[get, set\]](#)
Gets or sets whether the optional content group UI element is enabled or not.
- virtual bool [IsLocked \[get\]](#)
Indicates whether the state of the optional content group UI element is locked or can be changed.

7.34.1 Detailed Description

An optional content group UI element that can be enabled or disabled.

Definition at line [418](#) of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.34.2 Member Function Documentation

7.34.2.1 Toggle()

```
virtual void MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem.Toggle ( ) [virtual]
```

Toggle the state of the optional content group UI element.

Definition at line [445](#) of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.34.3 Property Documentation

7.34.3.1 IsEnabled

```
virtual bool MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem.IsEnabled [get], [set]
```

Gets or sets whether the optional content group UI element is enabled or not.

Definition at line [423](#) of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.34.3.2 IsLocked

```
virtual bool MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem.IsLocked [get]
```

Indicates whether the state of the optional content group UI element is locked or can be changed.

Definition at line 436 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

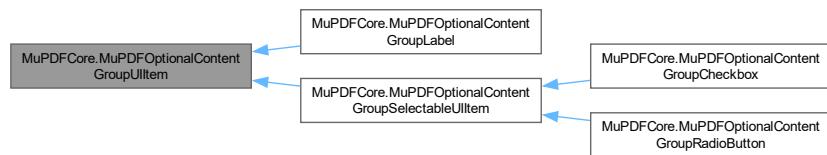
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.35 MuPDFCore.MuPDFOptionalContentGroupUIItem Class Reference

An optional content group UI element.

Inheritance diagram for MuPDFCore.MuPDFOptionalContentGroupUIItem:



Properties

- `virtual string Label [get]`
The label for this optional content group UI element.
- `virtual MuPDFOptionalContentGroupUIItem[] Children [get]`
Optional content group UI elements nested within this UI element (this may be empty, but it will not be null).

7.35.1 Detailed Description

An optional content group UI element.

Definition at line 396 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.35.2 Property Documentation

7.35.2.1 Children

```
virtual MuPDFOptionalContentGroupUIItem[] MuPDFCore.MuPDFOptionalContentGroupUIItem.Children  
[get]
```

Optional content group UI elements nested within this UI element (this may be empty, but it will not be null).

Definition at line 406 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

7.35.2.2 Label

```
virtual string MuPDFCore.MuPDFOptionalContentGroupUIItem.Label [get]
```

The label for this optional content group UI element.

Definition at line 401 of file [MuPDFOptionalContentGroupConfiguration.cs](#).

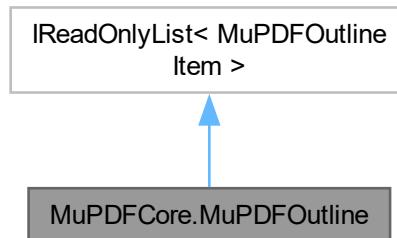
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs](#)

7.36 MuPDFCore.MuPDFOutline Class Reference

Represents a document outline (table of contents).

Inheritance diagram for MuPDFCore.MuPDFOutline:



Public Member Functions

- `IEnumerator< MuPDFOutlineItem > GetEnumerator ()`

Properties

- `IReadOnlyList< MuPDFOutlineItem > Items [get]`
The outline items.
- `int Count [get]`
- `MuPDFOutlineItem this[int index] [get]`

7.36.1 Detailed Description

Represents a document outline (table of contents).

Definition at line 30 of file [MuPDFOutline.cs](#).

7.36.2 Member Function Documentation

7.36.2.1 GetEnumerator()

```
IEnumerator< MuPDFOutlineItem > MuPDFCore.MuPDFOutline.GetEnumerator ( )
```

Definition at line 68 of file [MuPDFOutline.cs](#).

7.36.3 Property Documentation

7.36.3.1 Count

```
int MuPDFCore.MuPDFOutline.Count [get]
```

Definition at line 62 of file [MuPDFOutline.cs](#).

7.36.3.2 Items

```
IReadOnlyList<MuPDFOutlineItem> MuPDFCore.MuPDFOutline.Items [get]
```

The outline items.

Definition at line 35 of file [MuPDFOutline.cs](#).

7.36.3.3 this[int index]

`MuPDFOutlineItem MuPDFCore.MuPDFOutline.this[int index] [get]`

Definition at line 65 of file [MuPDFOutline.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFOutline.cs](#)

7.37 MuPDFCore.MuPDFOutlineItem Class Reference

Represents an item in a document outline (table of contents).

Properties

- string `Title` [get]
The title of the item. This can be null if the item does not have any text.
- string `Uri` [get]
The destination in the document to be displayed when this outline item is activated. This can be null if the item has no destination.
- int `Chapter` [get]
Locations within the document are referred to in terms of chapter and page, rather than just a page number. For some documents (such as epub documents with large numbers of pages broken into many chapters) this can make navigation much faster as only the required chapter needs to be decoded at a time.
- int `Page` [get]
The page number of an internal link, relative to the specified `Chapter`, or -1 for external links or links with no destination.
- int `PageNumber` [get]
The overall page number of an internal link. This is determined on first access, and it might cause a large number of chapters to be laid out to determine it.
- `PointF Location` [get]
The location on the page of the item pointed to by this outline item.
- `IReadOnlyList<MuPDFOutlineItem> Children` [get]
The sub items of this outline item (may be empty, but will not be null).

7.37.1 Detailed Description

Represents an item in a document outline (table of contents).

Definition at line 83 of file [MuPDFOutline.cs](#).

7.37.2 Property Documentation

7.37.2.1 Chapter

```
int MuPDFCore.MuPDFOutlineItem.Chapter [get]
```

Locations within the document are referred to in terms of chapter and page, rather than just a page number. For some documents (such as epub documents with large numbers of pages broken into many chapters) this can make navigation much faster as only the required chapter needs to be decoded at a time.

Definition at line 98 of file [MuPDFOutline.cs](#).

7.37.2.2 Children

```
IReadOnlyList<MuPDFOutlineItem> MuPDFCore.MuPDFOutlineItem.Children [get]
```

The sub items of this outline item (may be empty, but will not be null).

Definition at line 131 of file [MuPDFOutline.cs](#).

7.37.2.3 Location

```
PointF MuPDFCore.MuPDFOutlineItem.Location [get]
```

The location on the page of the item pointed to by this outline item.

Definition at line 126 of file [MuPDFOutline.cs](#).

7.37.2.4 Page

```
int MuPDFCore.MuPDFOutlineItem.Page [get]
```

The page number of an internal link, relative to the specified [Chapter](#), or -1 for external links or links with no destination.

Definition at line 103 of file [MuPDFOutline.cs](#).

7.37.2.5 PageNumber

```
int MuPDFCore.MuPDFOutlineItem.PageNumber [get]
```

The overall page number of an internal link. This is determined on first access, and it might cause a large number of chapters to be laid out to determine it.

Definition at line 110 of file [MuPDFOutline.cs](#).

7.37.2.6 Title

```
string MuPDFCore.MuPDFOutlineItem.Title [get]
```

The title of the item. This can be null if the item does not have any text.

Definition at line 88 of file [MuPDFOutline.cs](#).

7.37.2.7 Uri

```
string MuPDFCore.MuPDFOutlineItem.Uri [get]
```

The destination in the document to be displayed when this outline item is activated. This can be null if the item has no destination.

Definition at line 93 of file [MuPDFOutline.cs](#).

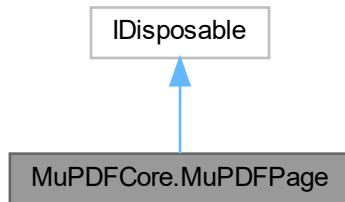
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFOutline.cs

7.38 MuPDFCore.MuPDFPage Class Reference

A wrapper over a [MuPDF](#) page object, which contains information about the page's boundaries.

Inheritance diagram for MuPDFCore.MuPDFPage:



Public Member Functions

- [Rectangle GetBoundingBox \(BoxType boxType, bool rescale=true\)](#)
Gets the specified bounding box from the page.
- [void Dispose \(\)](#)

Properties

- **Rectangle Bounds** [get]
The page's bounds at 72 DPI. Read-only.
- **int PageNumber** [get]
The number of this page in the original document.
- **MuPDFLinks Links** [get]
The links contained within the [MuPDFPage](#). This collection is populated on first access.

7.38.1 Detailed Description

A wrapper over a [MuPDF](#) page object, which contains information about the page's boundaries.

Definition at line 28 of file [MuPDFPage.cs](#).

7.38.2 Member Function Documentation

7.38.2.1 Dispose()

```
void MuPDFCore.MuPDFPage.Dispose ( )
```

Definition at line 194 of file [MuPDFPage.cs](#).

7.38.2.2 GetBoundingBox()

```
Rectangle MuPDFCore.MuPDFPage.GetBoundingBox ( 
    BoxType boxType,
    bool rescale = true )
```

Gets the specified bounding box from the page.

Parameters

<code>boxType</code>	The type of bounding box to get.
<code>rescale</code>	If this is <code>true</code> , the bounding box is rescaled so that it is expressed in the same resolution units as the underlying document. If this is <code>false</code> , the raw value is returned (at 72 dpi).

Returns

A [Rectangle](#) corresponding to the specified bounding box.

Exceptions

MuPDFException	Thrown if the bounding box cannot be computed or if another error occurs.
--------------------------------	---

Definition at line 130 of file [MuPDFPage.cs](#).

7.38.3 Property Documentation

7.38.3.1 Bounds

`Rectangle MuPDFCore.MuPDFPage.Bounds [get]`

The page's bounds at 72 DPI. Read-only.

Definition at line 33 of file [MuPDFPage.cs](#).

7.38.3.2 Links

`MuPDFLinks MuPDFCore.MuPDFPage.Links [get]`

The links contained within the [MuPDFPage](#). This collection is populated on first access.

Definition at line 45 of file [MuPDFPage.cs](#).

7.38.3.3 PageNumber

`int MuPDFCore.MuPDFPage.PageNumber [get]`

The number of this page in the original document.

Definition at line 38 of file [MuPDFPage.cs](#).

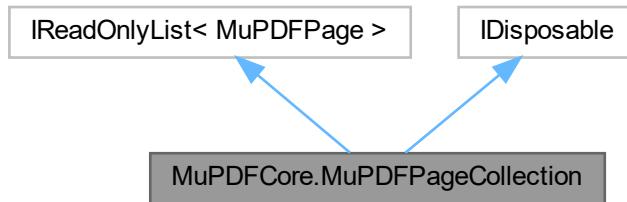
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFPage.cs

7.39 MuPDFCore.MuPDFPageCollection Class Reference

A lazy collection of [MuPDFPage](#)s. Each page is loaded from the document as it is requested for the first time.

Inheritance diagram for MuPDFCore.MuPDFPageCollection:



Public Member Functions

- `IEnumerator< MuPDFPage > GetEnumerator ()`
inherit doc
- `void Dispose ()`

Properties

- `int Length [get]`
The number of pages in the collection.
- `int Count [get]`
The number of pages in the collection.
- `MuPDFPage this[int index] [get]`
Get a page from the collection.

7.39.1 Detailed Description

A lazy collection of [MuPDFPage](#)s. Each page is loaded from the document as it is requested for the first time.

Definition at line 204 of file [MuPDFPage.cs](#).

7.39.2 Member Function Documentation

7.39.2.1 Dispose()

```
void MuPDFCore.MuPDFPageCollection.Dispose ( )
```

Definition at line 307 of file [MuPDFPage.cs](#).

7.39.2.2 GetEnumerator()

```
IEnumerator< MuPDFPage > MuPDFCore.MuPDFPageCollection.GetEnumerator ( )
```

inheritDoc/

Definition at line 268 of file [MuPDFPage.cs](#).

7.39.3 Property Documentation

7.39.3.1 Count

```
int MuPDFCore.MuPDFPageCollection.Count [get]
```

The number of pages in the collection.

Definition at line 229 of file [MuPDFPage.cs](#).

7.39.3.2 Length

```
int MuPDFCore.MuPDFPageCollection.Length [get]
```

The number of pages in the collection.

Definition at line 224 of file [MuPDFPage.cs](#).

7.39.3.3 this[int index]

```
MuPDFPage MuPDFCore.MuPDFPageCollection.this[int index] [get]
```

Get a page from the collection.

Parameters

<i>index</i>	The number of the page (starting at 0).
--------------	---

Returns

The specified [MuPDFPage](#).

Definition at line [236](#) of file [MuPDFPage.cs](#).

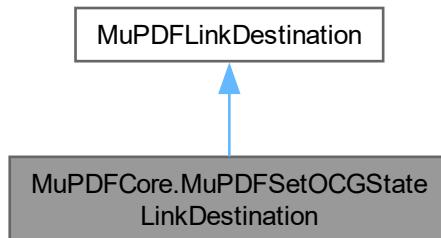
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFPage.cs

7.40 MuPDFCore.MuPDFSetOCGStateLinkDestination Class Reference

A destination for a link whose effect is to change the visibility state of some optional content groups (also known as layers).

Inheritance diagram for MuPDFCore.MuPDFSetOCGStateLinkDestination:



Public Member Functions

- void [Activate \(\)](#)
Activate the link destination, thus showing/hiding optional content groups as necessary.

Properties

- override [DestinationType Type](#) [get]

Additional Inherited Members

7.40.1 Detailed Description

A destination for a link whose effect is to change the visibility state of some optional content groups (also known as layers).

Definition at line 250 of file [MuPDFLinks.cs](#).

7.40.2 Member Function Documentation

7.40.2.1 Activate()

```
void MuPDFCore.MuPDFSetOCGStateLinkDestination.Activate ( )
```

Activate the link destination, thus showing/hiding optional content groups as necessary.

Definition at line 260 of file [MuPDFLinks.cs](#).

7.40.3 Property Documentation

7.40.3.1 Type

```
override DestinationType MuPDFCore.MuPDFSetOCGStateLinkDestination.Type [get]
```

Definition at line 253 of file [MuPDFLinks.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFLinks.cs

7.41 MuPDFCore.PDFCreationOptions Class Reference

Options for creating a PDF document.

Classes

- class [DocumentPermissions](#)

Specifies which actions can be performed without the owner password.

Public Types

- enum [CompressionOptions](#)
Stream compression options for PDF documents.
- enum [GarbageCollectionOption](#)
Options for garbage collection.
- enum [Encryption](#)
Document encryption options.

Properties

- bool [IncludeAnnotations](#) = true [get, set]
If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.
- [CompressionOptions CompressStreams](#) [get, set]
Determines whether streams are compressed in the generated PDF document.
- bool [AsciiEncode](#) = false [get, set]
Use ASCII hex encoding for binary streams.
- bool [PrettyPrint](#) = false [get, set]
Pretty-print objects with indentation.
- bool [Linearize](#) = false [get, set]
Generate a linearized PDF, optimized for loading in web browsers.
- bool [Clean](#) = false [get, set]
Pretty-print graphics commands in content streams.
- bool [Sanitize](#) = false [get, set]
Sanitize graphics commands in content streams.
- [GarbageCollectionOption Garbage](#) = GarbageCollectionOption.None [get, set]
Garbage collection options.
- [Encryption EncryptionType](#) = Encryption.None [get, set]
Type of encryption to use.
- string [UserPassword](#) = null [get, set]
User password required to read the newly created document.
- string [OwnerPassword](#) = null [get, set]
Owner password required to edit the newly created document.
- [DocumentPermissions Permissions](#) = null [get, set]
Actions that can be performed without the owner password (an OwnerPassword must be supplied, and an EncryptionType must be specified).
- bool [RegenerateId](#) = true [get, set]
Regenerate document id.

7.41.1 Detailed Description

Options for creating a PDF document.

Definition at line 13 of file [MuPDFDocument.Create.cs](#).

7.41.2 Member Enumeration Documentation

7.41.2.1 CompressionOptions

```
enum MuPDFCore.PDFCreationOptions.CompressionOptions
```

Stream compression options for PDF documents.

Definition at line 18 of file [MuPDFDocument.Create.cs](#).

7.41.2.2 Encryption

```
enum MuPDFCore.PDFCreationOptions.Encryption
```

Document encryption options.

Definition at line 75 of file [MuPDFDocument.Create.cs](#).

7.41.2.3 GarbageCollectionOption

```
enum MuPDFCore.PDFCreationOptions.GarbageCollectionOption
```

Options for garbage collection.

Definition at line 49 of file [MuPDFDocument.Create.cs](#).

7.41.3 Property Documentation

7.41.3.1 AsciiEncode

```
bool MuPDFCore.PDFCreationOptions.AsciiEncode = false [get], [set]
```

Use ASCII hex encoding for binary streams.

Definition at line 277 of file [MuPDFDocument.Create.cs](#).

7.41.3.2 Clean

```
bool MuPDFCore.PDFCreationOptions.Clean = false [get], [set]
```

Pretty-print graphics commands in content streams.

Definition at line 292 of file [MuPDFDocument.Create.cs](#).

7.41.3.3 CompressStreams

```
CompressionOptions MuPDFCore.PDFCreationOptions.CompressStreams [get], [set]
```

Determines whether streams are compressed in the generated PDF document.

Definition at line 260 of file [MuPDFDocument.Create.cs](#).

7.41.3.4 EncryptionType

```
Encryption MuPDFCore.PDFCreationOptions.EncryptionType = Encryption.None [get], [set]
```

Type of encryption to use.

Definition at line 307 of file [MuPDFDocument.Create.cs](#).

7.41.3.5 Garbage

```
GarbageCollectionOption MuPDFCore.PDFCreationOptions.Garbage = GarbageCollectionOption.None [get], [set]
```

Garbage collection options.

Definition at line 302 of file [MuPDFDocument.Create.cs](#).

7.41.3.6 IncludeAnnotations

```
bool MuPDFCore.PDFCreationOptions.IncludeAnnotations = true [get], [set]
```

If this is `true`, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.

Definition at line 253 of file [MuPDFDocument.Create.cs](#).

7.41.3.7 Linearize

```
bool MuPDFCore.PDFCreationOptions.Linearize = false [get], [set]
```

Generate a linearized PDF, optimized for loading in web browsers.

Definition at line 287 of file [MuPDFDocument.Create.cs](#).

7.41.3.8 OwnerPassword

```
string MuPDFCore.PDFCreationOptions.OwnerPassword = null [get], [set]
```

Owner password required to edit the newly created document.

Definition at line 317 of file [MuPDFDocument.Create.cs](#).

7.41.3.9 Permissions

```
DocumentPermissions MuPDFCore.PDFCreationOptions.Permissions = null [get], [set]
```

Actions that can be performed without the owner password (an [OwnerPassword](#) must be supplied, and an [EncryptionType](#) must be specified).

Definition at line 322 of file [MuPDFDocument.Create.cs](#).

7.41.3.10 PrettyPrint

```
bool MuPDFCore.PDFCreationOptions.PrettyPrint = false [get], [set]
```

Pretty-print objects with indentation.

Definition at line 282 of file [MuPDFDocument.Create.cs](#).

7.41.3.11 RegenerateId

```
bool MuPDFCore.PDFCreationOptions.RegenerateId = true [get], [set]
```

Regenerate document id.

Definition at line 327 of file [MuPDFDocument.Create.cs](#).

7.41.3.12 Sanitize

```
bool MuPDFCore.PDFCreationOptions.Sanitize = false [get], [set]
```

Sanitize graphics commands in content streams.

Definition at line 297 of file [MuPDFDocument.Create.cs](#).

7.41.3.13 UserPassword

```
string MuPDFCore.PDFCreationOptions.UserPassword = null [get], [set]
```

User password required to read the newly created document.

Definition at line 312 of file [MuPDFDocument.Create.cs](#).

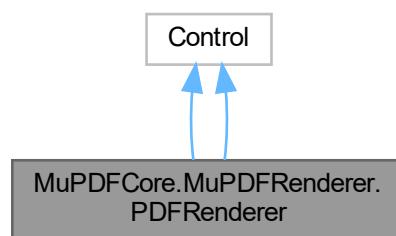
The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFDocument.Create.cs](#)

7.42 MuPDFCore.MuPDFRenderer.PDFRenderer Class Reference

A control to render PDF documents (and other formats), potentially using multiple threads.

Inheritance diagram for MuPDFCore.MuPDFRenderer.PDFRenderer:



Public Types

- enum [PointerEventHandlers](#)

Identifies the action to perform on pointer events.

Public Member Functions

- [PDFRenderer \(\)](#)

Initializes a new instance of the [PDFRenderer](#) class.

- void [Initialize \(\[MuPDFDocument\]\(#\) document, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, \[TesseractLanguage\]\(#\) ocrLanguage=null\)](#)

Set up the [PDFRenderer](#) to display a page of a [MuPDFDocument](#).

- async Task [InitializeAsync \(\[MuPDFDocument\]\(#\) document, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, \[TesseractLanguage\]\(#\) ocrLanguage=null, CancellationToken ocrCancellationToken=default, IProgress<\[OCRProgressInfo\]\(#\)> ocrProgress=null\)](#)

Set up the [PDFRenderer](#) to display a page of a [MuPDFDocument](#). The OCR step is run asynchronously, in order not to block the UI thread.

- void [Initialize](#) (string fileName, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from disk.
- async Task [InitializeAsync](#) (string fileName, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null, CancellationToken ocrCancellationToken=default, IProgress<OCRProgressInfo> ocrProgress=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from disk. The OCR step is run asynchronously, in order not to block the UI thread.
- void [Initialize](#) (MemoryStream ms, [InputFileTypes](#) fileType, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from a MemoryStream.
- async Task [InitializeAsync](#) (MemoryStream ms, [InputFileTypes](#) fileType, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null, CancellationToken ocrCancellationToken=default, IProgress<OCRProgressInfo> ocrProgress=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from a MemoryStream. The OCR step is run asynchronously, in order not to block the UI thread.
- void [Initialize](#) (byte[] dataBytes, [InputFileTypes](#) fileType, int offset=0, int length=-1, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from an array of bytes.
- async Task [InitializeAsync](#) (byte[] dataBytes, [InputFileTypes](#) fileType, int offset=0, int length=-1, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true, [TesseractLanguage](#) ocrLanguage=null, CancellationToken ocrCancellationToken=default, IProgress<OCRProgressInfo> ocrProgress=null)

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from an array of bytes. The OCR step is run asynchronously, in order not to block the UI thread.
- void [ReleaseResources](#) ()

Release resources held by this [PDFRenderer](#). This is not an irreversible step: using one of the Initialize overloads after calling this method will restore functionality.
- void [SetDisplayAreaNow](#) (Rect value)

Set the current display area to the specified value , skipping all transitions.
- void [ZoomStep](#) (double count, Point? center=null)

Zoom around a point.
- void [Contain](#) ()

Alter the display area so that the whole page fits on screen.
- void [Cover](#) ()

Alter the display area so that the page covers the whole surface of the [PDFRenderer](#) (even though parts of the page may be outside it).
- [RenderProgress GetProgress](#) ()

Get the current rendering progress.
- string [GetSelectedText](#) ()

Get the currently selected text.
- void [SelectAll](#) ()

Selects all the text in the document.
- int [Search](#) (Regex needle)

Highlights all matches of the specified Regex in the text and returns the number of matches found. Matches cannot span multiple lines.
- override void [Render](#) (DrawingContext context)

Draw the rendered document.

Static Public Attributes

- static readonly DirectProperty< PDFRenderer, int > **RenderThreadCountProperty** = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(RenderThreadCount), o => o.RenderThreadCount)
Defines the `RenderThreadCount` property.
- static readonly DirectProperty< PDFRenderer, int > **PageNumberProperty** = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(PageNumber), o => o.PageNumber)
Defines the `PageNumber` property.
- static readonly DirectProperty< PDFRenderer, bool > **IsViewerInitializedProperty** = AvaloniaProperty.RegisterDirect<PDFRenderer, bool>(nameof(IsViewerInitialized), o => o.IsViewerInitialized)
Defines the `IsViewerInitialized` property.
- static readonly DirectProperty< PDFRenderer, Rect > **PageSizeProperty** = AvaloniaProperty.RegisterDirect<PDFRenderer, Rect>(nameof(PageSize), o => o.PageSize)
Defines the `PageSize` property.
- static readonly StyledProperty< Rect > **DisplayAreaProperty** = AvaloniaProperty.Register<PDFRenderer, Rect>(nameof(DisplayArea))
Defines the `DisplayArea` property.
- static readonly StyledProperty< double > **ZoomIncrementProperty** = AvaloniaProperty.Register<PDFRenderer, double>(nameof(ZoomIncrement), Math.Pow(2, 1.0 / 3.0), defaultBindingMode: Avalonia.Data.BindingMode.TwoWay)
Defines the `ZoomIncrement` property.
- static readonly StyledProperty< IBrush > **BackgroundProperty** = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(Background))
Defines the `Background` property.
- static readonly StyledProperty< IBrush > **PageBackgroundProperty** = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(PageBackground))
Defines the `PageBackground` property.
- static readonly DirectProperty< PDFRenderer, double > **ZoomProperty** = AvaloniaProperty.RegisterDirect<PDFRenderer, double>(nameof(Zoom), o => o.Zoom, (o, v) => o.Zoom = v, defaultBindingMode: Avalonia.Data.BindingMode.TwoWay)
Defines the `Zoom` property.
- static readonly StyledProperty< PointerEventHandlers > **PointerEventHandlerTypeProperty** = AvaloniaProperty.Register<PDFRenderer, PointerEventHandlers>(nameof(PointerEventHandlersType), PointerEventHandlers.PanHighlight)
Defines the `PointerEventHandlersType` property.
- static readonly StyledProperty< bool > **ZoomEnabledProperty** = AvaloniaProperty.Register<PDFRenderer, bool>(nameof(ZoomEnabled), true)
Defines the `ZoomEnabled` property.
- static readonly StyledProperty< MuPDFStructuredTextAddressSpan > **SelectionProperty** = AvaloniaProperty.Register<PDFRenderer, MuPDFStructuredTextAddressSpan>(nameof(Selection), null)
Defines the `Selection` property.
- static readonly StyledProperty< IBrush > **SelectionBrushProperty** = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(SelectionBrush), new SolidColorBrush(Color.FromArgb(96, 86, 180, 233)))
Defines the `SelectionBrush` property.
- static readonly StyledProperty< IEnumerable< MuPDFStructuredTextAddressSpan > > **HighlightedRegionsProperty** = AvaloniaProperty.Register<PDFRenderer, IEnumerable<MuPDFStructuredTextAddressSpan>>(nameof(HighlightedRegions), null)
Defines the `HighlightedRegions` property.
- static readonly StyledProperty< IBrush > **HighlightBrushProperty** = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(HighlightBrush), new SolidColorBrush(Color.FromArgb(96, 230, 159, 0)))
Defines the `HighlightBrush` property.
- static readonly StyledProperty< IBrush > **LinkBrushProperty** = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(LinkBrush), null)

- Defines the [LinkBrush](#) property.*
- static readonly StyledProperty< IPen > [LinkPenProperty](#) = AvaloniaProperty.Register<PDFRenderer, IPen>(nameof([LinkPen](#)), new Pen(Color.FromRgb(0, 0, 255).ToInt32()))
Defines the [LinkPen](#) property.
 - static readonly StyledProperty< bool > [DrawLinksProperty](#) = AvaloniaProperty.Register<PDFRenderer, bool>(nameof([DrawLinks](#)), true)
Defines the [DrawLinks](#) property.
 - static readonly StyledProperty< bool > [ActivateLinksProperty](#) = AvaloniaProperty.Register<PDFRenderer, bool>(nameof([ActivateLinks](#)), true)
Defines the [ActivateLinks](#) property.

Properties

- int [RenderThreadCount](#) [get]
Exposes the number of threads that the current instance is using to render the document. Read-only.
- int [PageNumber](#) [get]
Exposes the number of the page that the current instance is rendering. Read-only.
- bool [IsViewerInitialized](#) [get]
Whether the current instance has been initialised with a document to render or not. Read-only.
- Rect [PageSize](#) [get]
Exposes the size of the page that is drawn by the current instance (in page units).
- Rect [DisplayArea](#) [get, set]
The region of the page (in page units) that is currently displayed by the current instance. This always has the same aspect ratio of the bounds of this control. When this is set, the value is sanitised so that the smallest rectangle with the correct aspect ratio containing the requested value is chosen.
- double [ZoomIncrement](#) [get, set]
Determines by how much the scale will be increased/decreased by the [ZoomStep\(double, Point?\)](#) method. Set this to a value smaller than 1 to invert the zoom in/out direction.
- IBrush [Background](#) [get, set]
The background colour of the control.
- IBrush [PageBackground](#) [get, set]
The background colour to use for the page drawn by the control.
- double [Zoom](#) [get, set]
The current zoom level. Setting this will change the [DisplayArea](#) appropriately, zooming around the center of the [DisplayArea](#).
- PointerEventHandlers [PointerEventHandlersType](#) [get, set]
Determines which default handlers for pointer events (which are used for panning around the page) should be enabled. If this is [PointerEventHandlers.Custom](#), you will have to implement your own way to pan around the document by changing the [DisplayArea](#).
- bool [ZoomEnabled](#) [get, set]
Whether the default handlers for pointer wheel events (which are used for zooming in/out) should be enabled. If this is false, you will have to implement your own way to zoom by changing the [DisplayArea](#).
- MuPDFStructuredTextAddressSpan [Selection](#) [get, set]
The start and end of the currently selected text.
- IBrush [SelectionBrush](#) [get, set]
The colour used to highlight the [Selection](#).
- IEnumerable< MuPDFStructuredTextAddressSpan > [HighlightedRegions](#) [get, set]
A collection of highlighted regions, e.g. as a result of a text search.
- IBrush [HighlightBrush](#) [get, set]
The colour used to highlight the [HighlightedRegions](#).
- IBrush [LinkBrush](#) [get, set]
The colour used to highlight links in the document.

- IPen [LinkPen](#) [get, set]
The colour used to highlight links in the document.
- bool [DrawLinks](#) [get, set]
Determines whether links are highlighted on the document.
- bool [ActivateLinks](#) [get, set]
Determines whether links on the document can be clicked.

Events

- EventHandler< [MupdfLinkClickedEventArgs](#) > [LinkClicked](#)
Fired when the user clicks on a link within the document.

7.42.1 Detailed Description

A control to render PDF documents (and other formats), potentially using multiple threads.

Definition at line 43 of file [PDFRenderer.cs](#).

7.42.2 Member Enumeration Documentation

7.42.2.1 PointerEventHandlers

```
enum MuPDFCore.MuPDFRenderer.PDFRenderer.PointerEventHandlers
```

Identifies the action to perform on pointer events.

Definition at line 246 of file [PDFRenderer.Properties.cs](#).

7.42.3 Constructor & Destructor Documentation

7.42.3.1 PDFRenderer()

```
MuPDFCore.MuPDFRenderer.PDFRenderer.PDFRenderer ( )
```

Initializes a new instance of the [PDFRenderer](#) class.

Definition at line 204 of file [PDFRenderer.cs](#).

7.42.4 Member Function Documentation

7.42.4.1 Contain()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Contain ( )
```

Alter the display area so that the whole page fits on screen.

Definition at line 714 of file [PDFRenderer.cs](#).

7.42.4.2 Cover()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Cover ( )
```

Alter the display area so that the page covers the whole surface of the [PDFRenderer](#) (even though parts of the page may be outside it).

Definition at line 723 of file [PDFRenderer.cs](#).

7.42.4.3 GetProgress()

```
RenderProgress MuPDFCore.MuPDFRenderer.PDFRenderer.GetProgress ( )
```

Get the current rendering progress.

Returns

A [RenderProgress](#) object with information about the rendering progress of each thread.

Definition at line 744 of file [PDFRenderer.cs](#).

7.42.4.4 GetSelectedText()

```
string MuPDFCore.MuPDFRenderer.PDFRenderer.GetSelectedText ( )
```

Get the currently selected text.

Returns

The currently selected text.

Definition at line 753 of file [PDFRenderer.cs](#).

7.42.4.5 Initialize() [1/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
    byte[] dataBytes,
    InputFileTypes fileType,
    int offset = 0,
    int length = -1,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from an array of bytes.

Parameters

<i>dataBytes</i>	The bytes of the document that should be opened. The array will be copied and can be safely discarded/alterred after this method returns.
<i>fileType</i>	The format of the document.
<i>offset</i>	The offset in the byte array at which the document starts.
<i>length</i>	The length of the document in bytes. If this is < 0, the whole array is used.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.

Definition at line 404 of file [PDFRenderer.cs](#).

7.42.4.6 Initialize() [2/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
    MemoryStream ms,
    InputFileTypes fileType,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from a [MemoryStream](#).

Parameters

<i>ms</i>	The MemoryStream containing the document that should be opened. This can be safely disposed after this method returns.
<i>fileType</i>	The format of the document.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.

Definition at line 360 of file [PDFRenderer.cs](#).

7.42.4.7 Initialize() [3/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
    MuPDFDocument document,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null )
```

Set up the [PDFRenderer](#) to display a page of a [MuPDFDocument](#).

Parameters

<i>document</i>	The MuPDFDocument to render.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.

Definition at line 259 of file [PDFRenderer.cs](#).

7.42.4.8 Initialize() [4/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
    string fileName,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from disk.

Parameters

<i>fileName</i>	The path to the document that should be opened.
-----------------	---

Parameters

<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.

Definition at line 309 of file [PDFRenderer.cs](#).

7.42.4.9 InitializeAsync() [1/4]

```
async Task MuPDFCore.MuPDFRenderer.PDFRenderer.InitializeAsync (
    byte[] dataBytes,
    InputFileType fileType,
    int offset = 0,
    int length = -1,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null,
    CancellationToken ocrCancellationToken = default,
    IProgress< OCRProgressInfo > ocrProgress = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from an array of bytes. The OCR step is run asynchronously, in order not to block the UI thread.

Parameters

<i>dataBytes</i>	The bytes of the document that should be opened. The array will be copied and can be safely discarded/altered after this method returns.
<i>fileType</i>	The format of the document.
<i>offset</i>	The offset in the byte array at which the document starts.
<i>length</i>	The length of the document in bytes. If this is < 0, the whole array is used.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.

Parameters

<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>ocrCancellationToken</i>	A CancellationToken used to cancel the OCR operation.
<i>ocrProgress</i>	An IProgress<OCRProgressInfo> used to report OCR progress.

Definition at line 447 of file [PDFRenderer.cs](#).

7.42.4.10 InitializeAsync() [2/4]

```
async Task MuPDFCore.MuPDFRenderer.PDFRenderer.InitializeAsync (
    MemoryStream ms,
    InputFileTypes fileType,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null,
    CancellationToken ocrCancellationToken = default,
    IProgress<OCRProgressInfo> ocrProgress = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from a [MemoryStream](#). The OCR step is run asynchronously, in order not to block the UI thread.

Parameters

<i>ms</i>	The MemoryStream containing the document that should be opened. This can be safely disposed after this method returns.
<i>fileType</i>	The format of the document.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is true, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>ocrCancellationToken</i>	A CancellationToken used to cancel the OCR operation.
<i>ocrProgress</i>	An IProgress<OCRProgressInfo> used to report OCR progress.

Definition at line 382 of file [PDFRenderer.cs](#).

7.42.4.11 InitializeAsync() [3/4]

```
async Task MuPDFCore.MuPDFRenderer.PDFRenderer.InitializeAsync (
    MuPDFDocument document,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null,
    CancellationToken ocrCancellationToken = default,
    IProgress< OCRProgressInfo > ocrProgress = null )
```

Set up the [PDFRenderer](#) to display a page of a [MuPDFDocument](#). The OCR step is run asynchronously, in order not to block the UI thread.

Parameters

<i>document</i>	The MuPDFDocument to render.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is <code>true</code> , annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is <code>null</code> , no OCR is performed.
<i>ocrCancellationToken</i>	A CancellationToken used to cancel the OCR operation.
<i>ocrProgress</i>	An IProgress<OCRProgressInfo> used to report OCR progress.

Definition at line 285 of file [PDFRenderer.cs](#).

7.42.4.12 InitializeAsync() [4/4]

```
async Task MuPDFCore.MuPDFRenderer.PDFRenderer.InitializeAsync (
    string fileName,
    int threadCount = 0,
    int pageNumber = 0,
    double resolutionMultiplier = 1,
    bool includeAnnotations = true,
    TesseractLanguage ocrLanguage = null,
    CancellationToken ocrCancellationToken = default,
    IProgress< OCRProgressInfo > ocrProgress = null )
```

Set up the [PDFRenderer](#) to display a page of a document that will be loaded from disk. The OCR step is run asynchronously, in order not to block the UI thread.

Parameters

<i>fileName</i>	The path to the document that should be opened.
<i>threadCount</i>	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than <i>threadCount</i> that satisfies this condition is used.
<i>pageNumber</i>	The index of the page that should be rendered. The first page has index 0.
<i>resolutionMultiplier</i>	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If <i>resolutionMultiplier</i> is 1, the resolution will match the size (in screen units) of the PDFRenderer .
<i>includeAnnotations</i>	If this is true, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.
<i>ocrLanguage</i>	The language to use for optical character recognition (OCR). If this is null, no OCR is performed.
<i>ocrCancellationToken</i>	A CancellationToken used to cancel the OCR operation.
<i>ocrProgress</i>	An IProgress<OCRProgressInfo> used to report OCR progress.

Definition at line 335 of file [PDFRenderer.cs](#).

7.42.4.13 ReleaseResources()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.ReleaseResources ( )
```

Release resources held by this [PDFRenderer](#). This is not an irreversible step: using one of the Initialize overloads after calling this method will restore functionality.

Definition at line 622 of file [PDFRenderer.cs](#).

7.42.4.14 Render()

```
override void MuPDFCore.MuPDFRenderer.PDFRenderer.Render (
    DrawingContext context )
```

Draw the rendered document.

Parameters

<i>context</i>	The drawing context on which to draw.
----------------	---------------------------------------

Definition at line 1377 of file [PDFRenderer.cs](#).

7.42.4.15 Search()

```
int MuPDFCore.MuPDFRenderer.PDFRenderer.Search (
    Regex needle )
```

Highlights all matches of the specified Regex in the text and returns the number of matches found. Matches cannot span multiple lines.

Parameters

<i>needle</i>	The Regex to search for.
---------------	--------------------------

Returns

The number of matches that have been found.

Definition at line [782](#) of file [PDFRenderer.cs](#).

7.42.4.16 SelectAll()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.SelectAll ( )
```

Selects all the text in the document.

Definition at line [761](#) of file [PDFRenderer.cs](#).

7.42.4.17 SetDisplayAreaNow()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.SetDisplayAreaNow (
    Rect value )
```

Set the current display area to the specified *value* , skipping all transitions.

Parameters

<i>value</i>	The new display area.
--------------	-----------------------

Definition at line [676](#) of file [PDFRenderer.cs](#).

7.42.4.18 ZoomStep()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomStep (
    double count,
    Point? center = null )
```

Zoom around a point.

Parameters

<i>count</i>	Number of steps to zoom. Positive values indicate a zoom in, negative values a zoom out.
<i>center</i>	The point around which to center the zoom operation. If this is null, the center of the control is used.

Definition at line 689 of file [PDFRenderer.cs](#).

7.42.5 Member Data Documentation

7.42.5.1 ActivateLinksProperty

```
readonly StyledProperty<bool> MuPDFCore.MuPDFRenderer.PDFRenderer.ActivateLinksProperty =  
AvaloniaProperty.Register<PDFRenderer, bool>(nameof(ActivateLinks), true) [static]
```

Defines the [ActivateLinks](#) property.

Definition at line 389 of file [PDFRenderer.Properties.cs](#).

7.42.5.2 BackgroundProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.BackgroundProperty =  
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(Background)) [static]
```

Defines the [Background](#) property.

Definition at line 182 of file [PDFRenderer.Properties.cs](#).

7.42.5.3 DisplayAreaProperty

```
readonly StyledProperty<Rect> MuPDFCore.MuPDFRenderer.PDFRenderer.DisplayAreaProperty = Avalonia←  
Property.Register<PDFRenderer, Rect>(nameof(DisplayArea)) [static]
```

Defines the [DisplayArea](#) property.

Definition at line 128 of file [PDFRenderer.Properties.cs](#).

7.42.5.4 DrawLinksProperty

```
readonly StyledProperty<bool> MuPDFCore.MuPDFRenderer.PDFRenderer.DrawLinksProperty = AvaloniaProperty.Register<PDFRenderer, bool>(nameof(DrawLinks), true) [static]
```

Defines the [DrawLinks](#) property.

Definition at line 376 of file [PDFRenderer.Properties.cs](#).

7.42.5.5 HighlightBrushProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.HighlightBrushProperty = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(HighlightBrush), new SolidColorBrush(Color.FromArgb(96, 230, 159, 0))) [static]
```

Defines the [HighlightBrush](#) property.

Definition at line 337 of file [PDFRenderer.Properties.cs](#).

7.42.5.6 HighlightedRegionsProperty

```
readonly StyledProperty<IEnumerable<MuPDFStructuredTextAddressSpan>> MuPDFCore.MuPDFRenderer.PDFRenderer.HighlightedRegionsProperty = AvaloniaProperty.Register<PDFRenderer, IEnumerable<MuPDFStructuredTextAddressSpan>>(nameof(HighlightedRegions), null) [static]
```

Defines the [HighlightedRegions](#) property.

Definition at line 324 of file [PDFRenderer.Properties.cs](#).

7.42.5.7 IsViewerInitializedProperty

```
readonly DirectProperty<PDFRenderer, bool> MuPDFCore.MuPDFRenderer.PDFRenderer.IsViewerInitializedProperty = AvaloniaProperty.RegisterDirect<PDFRenderer, bool>(nameof(IsViewerInitialized), o => o.IsViewerInitialized) [static]
```

Defines the [IsViewerInitialized](#) property.

Definition at line 80 of file [PDFRenderer.Properties.cs](#).

7.42.5.8 LinkBrushProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.LinkBrushProperty = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(LinkBrush), null) [static]
```

Defines the [LinkBrush](#) property.

Definition at line 350 of file [PDFRenderer.Properties.cs](#).

7.42.5.9 LinkPenProperty

```
readonly StyledProperty<IPen> MuPDFCore.MuPDFRenderer.PDFRenderer.LinkPenProperty = AvaloniaProperty.Register<PDFRenderer, IPen>(nameof(LinkPen), new Pen(Color.FromRgb(0, 0, 255).ToUInt32())) [static]
```

Defines the [LinkPen](#) property.

Definition at line 363 of file [PDFRenderer.Properties.cs](#).

7.42.5.10 PageBackgroundProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.PageBackgroundProperty = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(PageBackground)) [static]
```

Defines the [PageBackground](#) property.

Definition at line 195 of file [PDFRenderer.Properties.cs](#).

7.42.5.11 PageNumberProperty

```
readonly DirectProperty<PDFRenderer, int> MuPDFCore.MuPDFRenderer.PDFRenderer.PageNumberProperty = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(PageNumber), o => o.PageNumber) [static]
```

Defines the [PageNumber](#) property.

Definition at line 56 of file [PDFRenderer.Properties.cs](#).

7.42.5.12 PageSizeProperty

```
readonly DirectProperty<PDFRenderer, Rect> MuPDFCore.MuPDFRenderer.PDFRenderer.PageSize<-
Property = AvaloniaProperty.RegisterDirect<PDFRenderer, Rect>(nameof(PageSize), o => o.PageSize) [static]
```

Defines the [PageSize](#) property.

Definition at line 104 of file [PDFRenderer.Properties.cs](#).

7.42.5.13 PointerEventHandlerTypeProperty

```
readonly StyledProperty<PointerEventHandlers> MuPDFCore.MuPDFRenderer.PDFRenderer.Pointer-
EventHandlerTypeProperty = AvaloniaProperty.Register<PDFRenderer, PointerEventHandlers>(nameof(PointerEvent-
Handlers.PanHighlight)) [static]
```

Defines the [PointerEventHandlersType](#) property.

Definition at line 272 of file [PDFRenderer.Properties.cs](#).

7.42.5.14 RenderThreadCountProperty

```
readonly DirectProperty<PDFRenderer, int> MuPDFCore.MuPDFRenderer.PDFRenderer.RenderThread-
CountProperty = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(RenderThreadCount), o => o.RenderThreadCount) [static]
```

Defines the [RenderThreadCount](#) property.

Definition at line 32 of file [PDFRenderer.Properties.cs](#).

7.42.5.15 SelectionBrushProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.SelectionBrushProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(SelectionBrush), new SolidColorBrush(
Brush(Color.FromArgb(96, 86, 180, 233))) [static]
```

Defines the [SelectionBrush](#) property.

Definition at line 311 of file [PDFRenderer.Properties.cs](#).

7.42.5.16 SelectionProperty

```
readonly StyledProperty<MuPDFStructuredTextAddressSpan> MuPDFCore.MuPDFRenderer.PDFRenderer.←
SelectionProperty = AvaloniaProperty.Register<PDFRenderer, MuPDFStructuredTextAddressSpan>(nameof(Selection),
null) [static]
```

Defines the [Selection](#) property.

Definition at line [298](#) of file [PDFRenderer.Properties.cs](#).

7.42.5.17 ZoomEnabledProperty

```
readonly StyledProperty<bool> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomEnabledProperty = Avalonia←
Property.Register<PDFRenderer, bool>(nameof(ZoomEnabled), true) [static]
```

Defines the [ZoomEnabled](#) property.

Definition at line [285](#) of file [PDFRenderer.Properties.cs](#).

7.42.5.18 ZoomIncrementProperty

```
readonly StyledProperty<double> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomIncrementProperty =
AvaloniaProperty.Register<PDFRenderer, double>(nameof(ZoomIncrement), Math.Pow(2, 1.0 / 3.0),
defaultBindingMode: Avalonia.Data.BindingMode.TwoWay) [static]
```

Defines the [ZoomIncrement](#) property.

Definition at line [160](#) of file [PDFRenderer.Properties.cs](#).

7.42.5.19 ZoomProperty

```
readonly DirectProperty<PDFRenderer, double> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomProperty
= AvaloniaProperty.RegisterDirect<PDFRenderer, double>(nameof(Zoom), o => o.Zoom, (o, v) =>
o.Zoom = v, defaultBindingMode: Avalonia.Data.BindingMode.TwoWay) [static]
```

Defines the [Zoom](#) property.

Definition at line [208](#) of file [PDFRenderer.Properties.cs](#).

7.42.6 Property Documentation

7.42.6.1 ActivateLinks

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.ActivateLinks [get], [set]
```

Determines whether links on the document can be clicked.

Definition at line 393 of file [PDFRenderer.Properties.cs](#).

7.42.6.2 Background

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.Background [get], [set]
```

The background colour of the control.

Definition at line 186 of file [PDFRenderer.Properties.cs](#).

7.42.6.3 DisplayArea

```
Rect MuPDFCore.MuPDFRenderer.PDFRenderer.DisplayArea [get], [set]
```

The region of the page (in page units) that is currently displayed by the current instance. This always has the same aspect ratio of the bounds of this control. When this is set, the value is sanitised so that the smallest rectangle with the correct aspect ratio containing the requested value is chosen.

Definition at line 133 of file [PDFRenderer.Properties.cs](#).

7.42.6.4 DrawLinks

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.DrawLinks [get], [set]
```

Determines whether links are highlighted on the document.

Definition at line 380 of file [PDFRenderer.Properties.cs](#).

7.42.6.5 HighlightBrush

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.HighlightBrush [get], [set]
```

The colour used to highlight the [HighlightedRegions](#).

Definition at line 341 of file [PDFRenderer.Properties.cs](#).

7.42.6.6 **HighlightedRegions**

```
IEnumerable<MuPDFStructuredTextAddressSpan> MuPDFCore.MuPDFRenderer.PDFRenderer.HighlightedRegions [get], [set]
```

A collection of highlighted regions, e.g. as a result of a text search.

Definition at line 328 of file [PDFRenderer.Properties.cs](#).

7.42.6.7 **IsViewerInitialized**

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.IsViewerInitialized [get]
```

Whether the current instance has been initialised with a document to render or not. Read-only.

Definition at line 88 of file [PDFRenderer.Properties.cs](#).

7.42.6.8 **LinkBrush**

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.LinkBrush [get], [set]
```

The colour used to highlight links in the document.

Definition at line 354 of file [PDFRenderer.Properties.cs](#).

7.42.6.9 **LinkPen**

```
IPen MuPDFCore.MuPDFRenderer.PDFRenderer.LinkPen [get], [set]
```

The colour used to highlight links in the document.

Definition at line 367 of file [PDFRenderer.Properties.cs](#).

7.42.6.10 **PageBackground**

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.PageBackground [get], [set]
```

The background colour to use for the page drawn by the control.

Definition at line 199 of file [PDFRenderer.Properties.cs](#).

7.42.6.11 PageNumber

```
int MuPDFCore.MuPDFRenderer.PDFRenderer.PageNumber [get]
```

Exposes the number of the page that the current instance is rendering. Read-only.

Definition at line 64 of file [PDFRenderer.Properties.cs](#).

7.42.6.12 PageSize

```
Rect MuPDFCore.MuPDFRenderer.PDFRenderer.PageSize [get]
```

Exposes the size of the page that is drawn by the current instance (in page units).

Definition at line 112 of file [PDFRenderer.Properties.cs](#).

7.42.6.13 PointerEventHandlersType

```
PointerEventHandlers MuPDFCore.MuPDFRenderer.PDFRenderer.PointerEventHandlersType [get], [set]
```

Determines which default handlers for pointer events (which are used for panning around the page) should be enabled. If this is PointerEventHandlers.Custom, you will have to implement your own way to pan around the document by changing the [DisplayArea](#).

Definition at line 276 of file [PDFRenderer.Properties.cs](#).

7.42.6.14 RenderThreadCount

```
int MuPDFCore.MuPDFRenderer.PDFRenderer.RenderThreadCount [get]
```

Exposes the number of threads that the current instance is using to render the document. Read-only.

Definition at line 40 of file [PDFRenderer.Properties.cs](#).

7.42.6.15 Selection

```
MuPDFStructuredTextAddressSpan MuPDFCore.MuPDFRenderer.PDFRenderer.Selection [get], [set]
```

The start and end of the currently selected text.

Definition at line 302 of file [PDFRenderer.Properties.cs](#).

7.42.6.16 SelectionBrush

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.SelectionBrush [get], [set]
```

The colour used to highlight the [Selection](#).

Definition at line [315](#) of file [PDFRenderer.Properties.cs](#).

7.42.6.17 Zoom

```
double MuPDFCore.MuPDFRenderer.PDFRenderer.Zoom [get], [set]
```

The current zoom level. Setting this will change the [DisplayArea](#) appropriately, zooming around the center of the [DisplayArea](#).

Definition at line [216](#) of file [PDFRenderer.Properties.cs](#).

7.42.6.18 ZoomEnabled

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomEnabled [get], [set]
```

Whether the default handlers for pointer wheel events (which are used for zooming in/out) should be enabled. If this is false, you will have to implement your own way to zoom by changing the [DisplayArea](#).

Definition at line [289](#) of file [PDFRenderer.Properties.cs](#).

7.42.6.19 ZoomIncrement

```
double MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomIncrement [get], [set]
```

Determines by how much the scale will be increased/decreased by the [ZoomStep\(double, Point?\)](#) method. Set this to a value smaller than 1 to invert the zoom in/out direction.

Definition at line [164](#) of file [PDFRenderer.Properties.cs](#).

7.42.7 Event Documentation

7.42.7.1 LinkClicked

```
EventHandler<MuPDFLinkClickedEventArgs> MuPDFCore.MuPDFRenderer.PDFRenderer.LinkClicked
```

Fired when the user clicks on a link within the document.

Definition at line 402 of file [PDFRenderer.Properties.cs](#).

The documentation for this class was generated from the following files:

- MuPDFCore.MuPDFRenderer/PDFRenderer.cs
- MuPDFCore.MuPDFRenderer/PDFRenderer.Properties.cs

7.43 MuPDFCore.PointF Struct Reference

Represents a point.

Public Member Functions

- [PointF](#) (float x, float y)
Create a new [PointF](#) from the specified coordinates.

Public Attributes

- float [X](#)
The horizontal coordinate of the point.
- float [Y](#)
The vertical coordinate of the point.

7.43.1 Detailed Description

Represents a point.

Definition at line 566 of file [Rectangles.cs](#).

7.43.2 Constructor & Destructor Documentation

7.43.2.1 PointF()

```
MuPDFCore.PointF.PointF (
    float x,
    float y )
```

Create a new [PointF](#) from the specified coordinates.

Parameters

x	The horizontal coordinate of the point.
y	The vertical coordinate of the point.

Definition at line 583 of file [Rectangles.cs](#).

7.43.3 Member Data Documentation

7.43.3.1 X

```
float MuPDFCore.PointF.X
```

The horizontal coordinate of the point.

Definition at line 571 of file [Rectangles.cs](#).

7.43.3.2 Y

```
float MuPDFCore.PointF.Y
```

The vertical coordinate of the point.

Definition at line 576 of file [Rectangles.cs](#).

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

7.44 MuPDFCore.Quad Struct Reference

Represents a quadrilateral (not necessarily a rectangle).

Public Member Functions

- [Quad \(PointF lowerLeft, PointF upperLeft, PointF upperRight, PointF lowerRight\)](#)
*Creates a new **Quad** from the specified points.*
- bool [Contains \(PointF point\)](#)
*Checks whether this **Quad** contains a **PointF**.*

Public Attributes

- **PointF LowerLeft**
The lower left point of the quadrilater.
- **PointF UpperLeft**
The upper left point of the quadrilater.
- **PointF UpperRight**
The upper right point of the quadrilater.
- **PointF LowerRight**
The lower right point of the quadrilater.

7.44.1 Detailed Description

Represents a quadrilater (not necessarily a rectangle).

Definition at line 593 of file [Rectangles.cs](#).

7.44.2 Constructor & Destructor Documentation

7.44.2.1 Quad()

```
MuPDFCore.Quad.Quad (
    PointF lowerLeft,
    PointF upperLeft,
    PointF upperRight,
    PointF lowerRight )
```

Creates a new [Quad](#) from the specified points.

Parameters

<i>lowerLeft</i>	The lower left point of the quadrilater.
<i>upperLeft</i>	The upper left point of the quadrilater.
<i>upperRight</i>	The upper right point of the quadrilater.
<i>lowerRight</i>	The lower right point of the quadrilater.

Definition at line 622 of file [Rectangles.cs](#).

7.44.3 Member Function Documentation

7.44.3.1 Contains()

```
bool MuPDFCore.Quad.Contains (
    PointF point )
```

Checks whether this [Quad](#) contains a [PointF](#).

Parameters

<code>point</code>	The PointF to check.
--------------------	--------------------------------------

Returns

A boolean value indicating whether this [Quad](#) contains the *point*.

Definition at line 635 of file [Rectangles.cs](#).

7.44.4 Member Data Documentation

7.44.4.1 LowerLeft

```
PointF MuPDFCore.Quad.LowerLeft
```

The lower left point of the quadrilater.

Definition at line 598 of file [Rectangles.cs](#).

7.44.4.2 LowerRight

```
PointF MuPDFCore.Quad.LowerRight
```

The lower right point of the quadrilater.

Definition at line 613 of file [Rectangles.cs](#).

7.44.4.3 UpperLeft

```
PointF MuPDFCore.Quad.UpperLeft
```

The upper left point of the quadrilater.

Definition at line 603 of file [Rectangles.cs](#).

7.44.4.4 UpperRight

`PointF` MuPDFCore.Quad.UpperRight

The upper right point of the quadrilateral.

Definition at line 608 of file [Rectangles.cs](#).

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

7.45 MuPDFCore.Rectangle Struct Reference

Represents a rectangle.

Public Member Functions

- [Rectangle](#) (float x0, float y0, float x1, float y1)
Create a new [Rectangle](#) from the specified coordinates.
- [Rectangle](#) (double x0, double y0, double x1, double y1)
Create a new [Rectangle](#) from the specified coordinates.
- [RoundedRectangle Round](#) ()
Round the rectangle's coordinates to the closest integers.
- [RoundedRectangle Round](#) (double zoom)
Round the rectangle's coordinates to the closest integers, applying the specified zoom factor.
- [Rectangle\[\] Split](#) (int divisions)
Split the rectangle into the specified number of [Rectangles](#).
- [Rectangle Intersect](#) ([Rectangle](#) other)
Compute the intersection between this [Rectangle](#) and another one.
- bool [Contains](#) ([Rectangle](#) other)
Checks whether this [Rectangle](#) contains another [Rectangle](#).
- bool [Contains](#) ([PointF](#) point)
Checks whether this [Rectangle](#) contains a [PointF](#).
- [Quad ToQuad](#) ()
Converts the [Rectangle](#) to a [Quad](#).

Public Attributes

- float [X0](#)
The left coordinate of the rectangle.
- float [Y0](#)
The top coordinate of the rectangle.
- float [X1](#)
The right coordinate of the rectangle.
- float [Y1](#)
The bottom coordinate of the rectangle.

Properties

- float [Width](#) [get]
The width of the rectangle.
- float [Height](#) [get]
The height of the rectangle.

7.45.1 Detailed Description

Represents a rectangle.

Definition at line [326](#) of file [Rectangles.cs](#).

7.45.2 Constructor & Destructor Documentation

7.45.2.1 Rectangle() [1/2]

```
MuPDFCore.Rectangle.Rectangle (
    float x0,
    float y0,
    float x1,
    float y1 )
```

Create a new [Rectangle](#) from the specified coordinates.

Parameters

x0	The left coordinate of the rectangle.
y0	The top coordinate of the rectangle.
x1	The right coordinate of the rectangle.
y1	The bottom coordinate of the rectangle.

Definition at line [365](#) of file [Rectangles.cs](#).

7.45.2.2 Rectangle() [2/2]

```
MuPDFCore.Rectangle.Rectangle (
    double x0,
    double y0,
    double x1,
    double y1 )
```

Create a new [Rectangle](#) from the specified coordinates.

Parameters

<code>x0</code>	The left coordinate of the rectangle.
<code>y0</code>	The top coordinate of the rectangle.
<code>x1</code>	The right coordinate of the rectangle.
<code>y1</code>	The bottom coordinate of the rectangle.

Definition at line 380 of file [Rectangles.cs](#).

7.45.3 Member Function Documentation

7.45.3.1 Contains() [1/2]

```
bool MuPDFCore.Rectangle.Contains (
    PointF point )
```

Checks whether this [Rectangle](#) contains a [PointF](#).

Parameters

<code>point</code>	The PointF to check.
--------------------	--------------------------------------

Returns

A boolean value indicating whether this [Rectangle](#) contains the `point`.

Definition at line 476 of file [Rectangles.cs](#).

7.45.3.2 Contains() [2/2]

```
bool MuPDFCore.Rectangle.Contains (
    Rectangle other )
```

Checks whether this [Rectangle](#) contains another [Rectangle](#).

Parameters

<code>other</code>	The Rectangle to check.
--------------------	---

Returns

A boolean value indicating whether this [Rectangle](#) contains the `other` [Rectangle](#).

Definition at line 466 of file [Rectangles.cs](#).

7.45.3.3 Intersect()

```
Rectangle MuPDFCore.Rectangle.Intersect (
    Rectangle other )
```

Compute the intersection between this [Rectangle](#) and another one.

Parameters

<i>other</i>	The other Rectangle to intersect with this instance.
--------------	--

Returns

The intersection between the two [Rectangles](#).

Definition at line 443 of file [Rectangles.cs](#).

7.45.3.4 Round() [1/2]

```
RoundedRectangle MuPDFCore.Rectangle.Round ( )
```

Round the rectangle's coordinates to the closest integers.

Returns

A [RoundedRectangle](#) with the rounded coordinates.

Definition at line 392 of file [Rectangles.cs](#).

7.45.3.5 Round() [2/2]

```
RoundedRectangle MuPDFCore.Rectangle.Round (
    double zoom )
```

Round the rectangle's coordinates to the closest integers, applying the specified zoom factor.

Parameters

<i>zoom</i>	The zoom factor to apply.
-------------	---------------------------

Returns

A [RoundedRectangle](#) with the rounded coordinates.

Definition at line 407 of file [Rectangles.cs](#).

7.45.3.6 Split()

```
Rectangle[ ] MuPDFCore.Rectangle.Split (
    int divisions )
```

Split the rectangle into the specified number of [Rectangles](#).

Parameters

<i>divisions</i>	The number of rectangles in which the rectangle should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <i>divisions</i> that satisfies this condition is used.
------------------	--

Returns

An array of [Rectangles](#) that when positioned properly cover the same area as this object.

Definition at line 422 of file [Rectangles.cs](#).

7.45.3.7 ToQuad()

```
Quad MuPDFCore.Rectangle.ToQuad ( )
```

Converts the [Rectangle](#) to a [Quad](#).

Returns

A [Quad](#) corresponding to the current [Rectangle](#).

Definition at line 485 of file [Rectangles.cs](#).

7.45.4 Member Data Documentation

7.45.4.1 X0

```
float MuPDFCore.Rectangle.X0
```

The left coordinate of the rectangle.

Definition at line 331 of file [Rectangles.cs](#).

7.45.4.2 X1

```
float MuPDFCore.Rectangle.X1
```

The right coordinate of the rectangle.

Definition at line 341 of file [Rectangles.cs](#).

7.45.4.3 Y0

```
float MuPDFCore.Rectangle.Y0
```

The top coordinate of the rectangle.

Definition at line 336 of file [Rectangles.cs](#).

7.45.4.4 Y1

```
float MuPDFCore.Rectangle.Y1
```

The bottom coordinate of the rectangle.

Definition at line 346 of file [Rectangles.cs](#).

7.45.5 Property Documentation

7.45.5.1 Height

```
float MuPDFCore.Rectangle.Height [get]
```

The height of the rectangle.

Definition at line 356 of file [Rectangles.cs](#).

7.45.5.2 Width

```
float MuPDFCore.Rectangle.Width [get]
```

The width of the rectangle.

Definition at line 351 of file [Rectangles.cs](#).

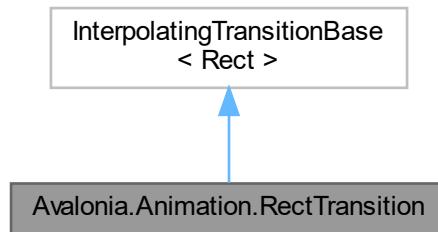
The documentation for this struct was generated from the following file:

- [MuPDFCore/Rectangles.cs](#)

7.46 Avalonia.Animation.RectTransition Class Reference

Transition class that handles AvaloniaProperty with Rect types.

Inheritance diagram for Avalonia.Animation.RectTransition:



7.46.1 Detailed Description

Transition class that handles AvaloniaProperty with Rect types.

Definition at line 23 of file [RectTransition.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore.MuPDFRenderer/RectTransition.cs](#)

7.47 MuPDFCore.RenderProgress Class Reference

Holds a summary of the progress of the current rendering operation.

Classes

- struct [ThreadRenderProgress](#)
Holds the progress of a single thread.

Properties

- [ThreadRenderProgress\[\] ThreadRenderProgresses](#) [get]
Contains the progress of all the threads used in rendering the document.

7.47.1 Detailed Description

Holds a summary of the progress of the current rendering operation.

Definition at line [485](#) of file [MuPDF.cs](#).

7.47.2 Property Documentation

7.47.2.1 ThreadRenderProgresses

[ThreadRenderProgress](#) [] MuPDFCore.RenderProgress.ThreadRenderProgresses [get]

Contains the progress of all the threads used in rendering the document.

Definition at line [512](#) of file [MuPDF.cs](#).

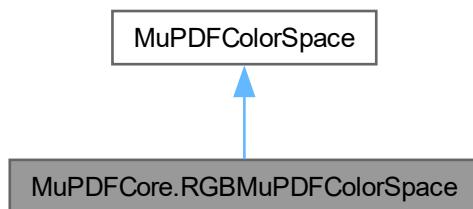
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

7.48 MuPDFCore.RGBMuPDFColorSpace Class Reference

Represents a colour space where each pixel is stored as three bytes encoding the R, G, and B components.

Inheritance diagram for MuPDFCore.RGBMuPDFColorSpace:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]

Additional Inherited Members

7.48.1 Detailed Description

Represents a colour space where each pixel is stored as three bytes encoding the R, G, and B components.

Definition at line 256 of file [MuPDFColorSpace.cs](#).

7.48.2 Property Documentation

7.48.2.1 NumBytes

```
override int MuPDFCore.RGBMuPDFColorSpace.NumBytes [get]
```

Definition at line 262 of file [MuPDFColorSpace.cs](#).

7.48.2.2 Type

```
override ColorSpaceType MuPDFCore.RGBMuPDFColorSpace.Type [get]
```

Definition at line 259 of file [MuPDFColorSpace.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFColorSpace.cs

7.49 MuPDFCore.RoundedRectangle Struct Reference

Represents a rectangle using only integer numbers.

Public Member Functions

- [RoundedRectangle](#) (int x0, int y0, int x1, int y1)
Create a new [RoundedRectangle](#) from the specified coordinates.
- [RoundedRectangle\[\] Split](#) (int divisions)
Split the rectangle into the specified number of [RoundedRectangles](#).

Public Attributes

- int **X0**
The left coordinate of the rectangle.
- int **Y0**
The top coordinate of the rectangle.
- int **X1**
The right coordinate of the rectangle.
- int **Y1**
The bottom coordinate of the rectangle.

Properties

- int **Width** [get]
The width of the rectangle.
- int **Height** [get]
The height of the rectangle.

7.49.1 Detailed Description

Represents a rectangle using only integer numbers.

Definition at line 494 of file [Rectangles.cs](#).

7.49.2 Constructor & Destructor Documentation

7.49.2.1 RoundedRectangle()

```
MuPDFCore.RoundedRectangle.RoundedRectangle (
    int x0,
    int y0,
    int x1,
    int y1 )
```

Create a new **RoundedRectangle** from the specified coordinates.

Parameters

x0	The left coordinate of the rectangle.
y0	The top coordinate of the rectangle.
x1	The right coordinate of the rectangle.
y1	The bottom coordinate of the rectangle.

Definition at line 533 of file [Rectangles.cs](#).

7.49.3 Member Function Documentation

7.49.3.1 Split()

```
RoundedRectangle[ ] MuPDFCore.RoundedRectangle.Split (
    int divisions )
```

Split the rectangle into the specified number of [RoundedRectangles](#).

Parameters

<i>divisions</i>	The number of rectangles in which the rectangle should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <i>divisions</i> that satisfies this condition is used.
------------------	--

Returns

An array of [RoundedRectangles](#) that when positioned properly cover the same area as this object.

Definition at line 546 of file [Rectangles.cs](#).

7.49.4 Member Data Documentation

7.49.4.1 X0

```
int MuPDFCore.RoundedRectangle.X0
```

The left coordinate of the rectangle.

Definition at line 499 of file [Rectangles.cs](#).

7.49.4.2 X1

```
int MuPDFCore.RoundedRectangle.X1
```

The right coordinate of the rectangle.

Definition at line 509 of file [Rectangles.cs](#).

7.49.4.3 Y0

```
int MuPDFCore.RoundedRectangle.Y0
```

The top coordinate of the rectangle.

Definition at line 504 of file [Rectangles.cs](#).

7.49.4.4 Y1

```
int MuPDFCore.RoundedRectangle.Y1
```

The bottom coordinate of the rectangle.

Definition at line 514 of file [Rectangles.cs](#).

7.49.5 Property Documentation

7.49.5.1 Height

```
int MuPDFCore.RoundedRectangle.Height [get]
```

The height of the rectangle.

Definition at line 524 of file [Rectangles.cs](#).

7.49.5.2 Width

```
int MuPDFCore.RoundedRectangle.Width [get]
```

The width of the rectangle.

Definition at line 519 of file [Rectangles.cs](#).

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

7.50 MuPDFCore.RoundedSize Struct Reference

Represents the size of a rectangle using only integer numbers.

Public Member Functions

- [RoundedSize](#) (int width, int height)
Create a new [RoundedSize](#) with the specified width and height.
- [RoundedRectangle\[\] Split](#) (int divisions)
Split the size into the specified number of [RoundedRectangles](#).

Public Attributes

- int [Width](#)
The width of the rectangle.
- int [Height](#)
The height of the rectangle.

7.50.1 Detailed Description

Represents the size of a rectangle using only integer numbers.

Definition at line 181 of file [Rectangles.cs](#).

7.50.2 Constructor & Destructor Documentation

7.50.2.1 [RoundedSize\(\)](#)

```
MuPDFCore.RoundedSize.RoundedSize (
    int width,
    int height )
```

Create a new [RoundedSize](#) with the specified width and height.

Parameters

<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.

Definition at line 198 of file [Rectangles.cs](#).

7.50.3 Member Function Documentation

7.50.3.1 Split()

```
RoundedRectangle[ ] MuPDFCore.RoundedSize.Split (
    int divisions )
```

Split the size into the specified number of [RoundedRectangles](#).

Parameters

<i>divisions</i>	The number of rectangles in which the size should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <i>divisions</i> that satisfies this condition is used.
------------------	---

Returns

An array of [RoundedRectangles](#) that when positioned properly cover an area of the size of this object.

Definition at line 209 of file [Rectangles.cs](#).

7.50.4 Member Data Documentation

7.50.4.1 Height

```
int MuPDFCore.RoundedSize.Height
```

The height of the rectangle.

Definition at line 191 of file [Rectangles.cs](#).

7.50.4.2 Width

```
int MuPDFCore.RoundedSize.Width
```

The width of the rectangle.

Definition at line 186 of file [Rectangles.cs](#).

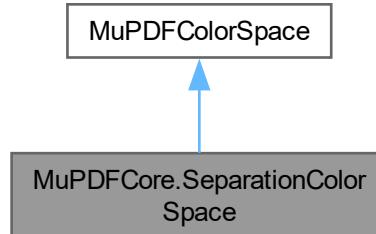
The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

7.51 MuPDFCore.SeparationColorSpace Class Reference

Represents a separation colour space.

Inheritance diagram for MuPDFCore.SeparationColorSpace:



Properties

- override [ColorSpaceType Type](#) [get]
- override int [NumBytes](#) [get]
- string[] [ColorantNames](#) [get]
Names of the separation colorants.
- [MuPDFColorSpace AlternateColorSpace](#) [get]
Alternate colour space.

Additional Inherited Members

7.51.1 Detailed Description

Represents a separation colour space.

Definition at line 352 of file [MuPDFColorSpace.cs](#).

7.51.2 Property Documentation

7.51.2.1 AlternateColorSpace

[MuPDFColorSpace](#) [MuPDFCore.SeparationColorSpace.AlternateColorSpace](#) [get]

Alternate colour space.

Definition at line 368 of file [MuPDFColorSpace.cs](#).

7.51.2.2 ColorantNames

```
string [] MuPDFCore.SeparationColorSpace.ColorantNames [get]
```

Names of the separation colorants.

Definition at line 363 of file [MuPDFColorSpace.cs](#).

7.51.2.3 NumBytes

```
override int MuPDFCore.SeparationColorSpace.NumBytes [get]
```

Definition at line 358 of file [MuPDFColorSpace.cs](#).

7.51.2.4 Type

```
override ColorSpaceType MuPDFCore.SeparationColorSpace.Type [get]
```

Definition at line 355 of file [MuPDFColorSpace.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFColorSpace.cs](#)

7.52 MuPDFCore.Size Struct Reference

Represents the size of a rectangle.

Public Member Functions

- [Size](#) (float width, float height)
Create a new [Size](#) with the specified width and height.
- [Size](#) (double width, double height)
Create a new [Size](#) with the specified width and height.
- [Rectangle\[\] Split](#) (int divisions)
Split the size into the specified number of [Rectangles](#).

Public Attributes

- float [Width](#)
The width of the rectangle.
- float [Height](#)
The height of the rectangle.

7.52.1 Detailed Description

Represents the size of a rectangle.

Definition at line 25 of file [Rectangles.cs](#).

7.52.2 Constructor & Destructor Documentation

7.52.2.1 Size() [1/2]

```
MuPDFCore.Size.Size (
    float width,
    float height )
```

Create a new [Size](#) with the specified width and height.

Parameters

<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.

Definition at line 42 of file [Rectangles.cs](#).

7.52.2.2 Size() [2/2]

```
MuPDFCore.Size.Size (
    double width,
    double height )
```

Create a new [Size](#) with the specified width and height.

Parameters

<i>width</i>	The width of the rectangle.
<i>height</i>	The height of the rectangle.

Definition at line 53 of file [Rectangles.cs](#).

7.52.3 Member Function Documentation

7.52.3.1 Split()

```
Rectangle[ ] MuPDFCore.Size.Split (
    int divisions )
```

Split the size into the specified number of [Rectangles](#).

Parameters

<i>divisions</i>	The number of rectangles in which the size should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <i>divisions</i> that satisfies this condition is used.
------------------	---

Returns

An array of [Rectangles](#) that when positioned properly cover an area of the size of this object.

Definition at line 64 of file [Rectangles.cs](#).

7.52.4 Member Data Documentation

7.52.4.1 Height

```
float MuPDFCore.Size.Height
```

The height of the rectangle.

Definition at line 35 of file [Rectangles.cs](#).

7.52.4.2 Width

```
float MuPDFCore.Size.Width
```

The width of the rectangle.

Definition at line 30 of file [Rectangles.cs](#).

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

7.53 MuPDFCore.SVGCreationOptions Class Reference

Options for creating an SVG document.

Public Types

- enum [TextOption](#)
Ways in which text is represented in the SVG document.

Properties

- bool [IncludeAnnotations](#) = true [get, set]
If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.
- [TextOption TextRendering](#) = TextOption.TextAsPath [get, set]
Determines how text is represented in the SVG document.
- bool [ReuseImages](#) = true [get, set]
Whether to reuse images using <symbol> definitions.

7.53.1 Detailed Description

Options for creating an SVG document.

Definition at line 546 of file [MuPDFDocument.Create.cs](#).

7.53.2 Member Enumeration Documentation

7.53.2.1 [TextOption](#)

enum [MuPDFCore.SVGCreationOptions.TextOption](#)

Ways in which text is represented in the SVG document.

Definition at line 551 of file [MuPDFDocument.Create.cs](#).

7.53.3 Property Documentation

7.53.3.1 [IncludeAnnotations](#)

bool [MuPDFCore.SVGCreationOptions.IncludeAnnotations](#) = true [get], [set]

If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.

Definition at line 567 of file [MuPDFDocument.Create.cs](#).

7.53.3.2 ReuseImages

```
bool MuPDFCore.SVGCreationOptions.ReuseImages = true [get], [set]
```

Whether to reuse images using <symbol> definitions.

Definition at line 577 of file [MuPDFDocument.Create.cs](#).

7.53.3.3 TextRendering

```
TextOption MuPDFCore.SVGCreationOptions.TextRendering = TextOption.TextAsPath [get], [set]
```

Determines how text is represented in the SVG document.

Definition at line 572 of file [MuPDFDocument.Create.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFDocument.Create.cs

7.54 MuPDFCore.TesseractLanguage Class Reference

Represents a language used by Tesseract OCR.

Public Types

- enum [Fast](#)
Fast integer versions of trained models. These are models for a single language.
- enum [FastScripts](#)
Fast integer versions of trained models. These are models for a single script supporting one or more languages.
- enum [Best](#)
Best (most accurate) trained models. These are models for a single language.
- enum [BestScripts](#)
Best (most accurate) trained models. These are models for a single script supporting one or more languages.

Public Member Functions

- [TesseractLanguage \(string prefix, string language\)](#)
Create a new `TesseractLanguage` object using the provided prefix and language name, without processing them in any way.
- [TesseractLanguage \(string fileName\)](#)
Create a new `TesseractLanguage` object using the specified trained model data file.
- [TesseractLanguage \(Fast language, bool useAnyCached=false\)](#)
Create a new `TesseractLanguage` object using a fast integer version of a trained model for the specified language. The language file is downloaded from the `tesseract-ocr/tessdata_fast` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.
- [TesseractLanguage \(Best language, bool useAnyCached=false\)](#)
Create a new `TesseractLanguage` object using the best (most accurate) version of the trained model for the specified language. The language file is downloaded from the `tesseract-ocr/tessdata_best` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.
- [TesseractLanguage \(FastScripts script, bool useAnyCached=false\)](#)
Create a new `TesseractLanguage` object using a fast integer version of a trained model for the specified script. The language file is downloaded from the `tesseract-ocr/tessdata_fast` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.
- [TesseractLanguage \(BestScripts script, bool useAnyCached=false\)](#)
Create a new `TesseractLanguage` object using the best (most accurate) version of the trained model for the specified script. The language file is downloaded from the `tesseract-ocr/tessdata_best` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.

Properties

- string [Prefix \[get\]](#)
The name of the folder where the language file is located.
- string [Language \[get\]](#)
The name of the language. The Tesseract library will assume that the trained language data file can be found at `Prefix/Language.traineddata`.

7.54.1 Detailed Description

Represents a language used by Tesseract OCR.

Definition at line 13 of file [TesseractLanguage.cs](#).

7.54.2 Member Enumeration Documentation

7.54.2.1 Best

```
enum MuPDFCore.TesseractLanguage.Best
```

Best (most accurate) trained models. These are models for a single language.

Definition at line 690 of file [TesseractLanguage.cs](#).

7.54.2.2 BestScripts

enum [MuPDFCore.TesseractLanguage.BestScripts](#)

Best (most accurate) trained models. These are models for a single script supporting one or more languages.

Definition at line [1193](#) of file [TesseractLanguage.cs](#).

7.54.2.3 Fast

enum [MuPDFCore.TesseractLanguage.Fast](#)

Fast integer versions of trained models. These are models for a single language.

Definition at line [28](#) of file [TesseractLanguage.cs](#).

7.54.2.4 FastScripts

enum [MuPDFCore.TesseractLanguage.FastScripts](#)

Fast integer versions of trained models. These are models for a single script supporting one or more languages.

Definition at line [535](#) of file [TesseractLanguage.cs](#).

7.54.3 Constructor & Destructor Documentation

7.54.3.1 TesseractLanguage() [1/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    string prefix,
    string language )
```

Create a new [TesseractLanguage](#) object using the provided *prefix* and *language* name, without processing them in any way.

Parameters

<i>prefix</i>	The name of the folder where the language file is located. If this is <code>null</code> , the value of the environment variable <code>TESSDATA_PREFIX</code> will be used.
<i>language</i>	The name of the language. The Tesseract library will assume that the trained language data file can be found at <i>prefix</i> / <i>language</i> <code>.traineddata</code> .

Definition at line 1350 of file [TesseractLanguage.cs](#).

7.54.3.2 TesseractLanguage() [2/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    string fileName )
```

Create a new [TesseractLanguage](#) object using the specified trained model data file.

Parameters

<i>fileName</i>	The path to the trained model data file. If the file name does not end in <code>.traineddata</code> , the file is copied to a temporary folder, and the temporary file is used by the Tesseract library.
-----------------	--

Definition at line 1360 of file [TesseractLanguage.cs](#).

7.54.3.3 TesseractLanguage() [3/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    Fast language,
    bool useAnyCached = false )
```

Create a new [TesseractLanguage](#) object using a fast integer version of a trained model for the specified language. The language file is downloaded from the `tesseract-ocr/tessdata_fast` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.

Parameters

<i>language</i>	The language to use for the OCR process.
<i>useAnyCached</i>	If this is <code>true</code> , if a cached trained model file is available for the specified language, it will be used even if it is a "best (most accurate)" model. Otherwise, only cached fast integer trained models will be used.

Definition at line 1387 of file [TesseractLanguage.cs](#).

7.54.3.4 TesseractLanguage() [4/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    Best language,
    bool useAnyCached = false )
```

Create a new [TesseractLanguage](#) object using the best (most accurate) version of the trained model for the specified language. The language file is downloaded from the `tesseract-ocr/tessdata_best` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.

Parameters

<i>language</i>	The language to use for the OCR process.
<i>useAnyCached</i>	If this is <code>true</code> , if a cached trained model file is available for the specified language, it will be used even if it is a "fast" model. Otherwise, only cached best (most accurate) trained models will be used.

Definition at line 1453 of file [TesseractLanguage.cs](#).

7.54.3.5 TesseractLanguage() [5/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    FastScripts script,
    bool useAnyCached = false )
```

Create a new [TesseractLanguage](#) object using a fast integer version of a trained model for the specified script. The language file is downloaded from the `tesseract-ocr/tessdata_fast` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.

Parameters

<i>script</i>	The script to use for the OCR process.
<i>useAnyCached</i>	If this is <code>true</code> , if a cached trained model file is available for the specified script, it will be used even if it is a "best (most accurate)" model. Otherwise, only cached fast integer trained models will be used.

Definition at line 1519 of file [TesseractLanguage.cs](#).

7.54.3.6 TesseractLanguage() [6/6]

```
MuPDFCore.TesseractLanguage.TesseractLanguage (
    BestScripts script,
    bool useAnyCached = false )
```

Create a new [TesseractLanguage](#) object using the best (most accurate) version of the trained model for the specified script. The language file is downloaded from the `tesseract-ocr/tessdata_best` GitHub repository. If it has already been downloaded and cached before, the downloaded file is re-used.

Parameters

<i>script</i>	The script to use for the OCR process.
<i>useAnyCached</i>	If this is <code>true</code> , if a cached trained model file is available for the specified script, it will be used even if it is a "fast" model. Otherwise, only cached best (most accurate) trained models will be used.

Definition at line 1589 of file [TesseractLanguage.cs](#).

7.54.4 Property Documentation

7.54.4.1 Language

```
string MuPDFCore.TesseractLanguage.Language [get]
```

The name of the language. The Tesseract library will assume that the trained language data file can be found at Prefix/Language.traineddata.

Definition at line 23 of file [TesseractLanguage.cs](#).

7.54.4.2 Prefix

```
string MuPDFCore.TesseractLanguage.Prefix [get]
```

The name of the folder where the language file is located.

Definition at line 18 of file [TesseractLanguage.cs](#).

The documentation for this class was generated from the following file:

- MuPDFCore/TesseractLanguage.cs

7.55 MuPDFCore.RenderProgress.ThreadRenderProgress Struct Reference

Holds the progress of a single thread.

Public Attributes

- int [Progress](#)
The current progress.
- long [MaxProgress](#)
The maximum progress. If this is 0, this value could not be determined (yet).

7.55.1 Detailed Description

Holds the progress of a single thread.

Definition at line 490 of file [MuPDF.cs](#).

7.55.2 Member Data Documentation

7.55.2.1 MaxProgress

```
long MuPDFCore.RenderProgress.ThreadRenderProgress.MaxProgress
```

The maximum progress. If this is 0, this value could not be determined (yet).

Definition at line 500 of file [MuPDF.cs](#).

7.55.2.2 Progress

```
int MuPDFCore.RenderProgress.ThreadRenderProgress.Progress
```

The current progress.

Definition at line 495 of file [MuPDF.cs](#).

The documentation for this struct was generated from the following file:

- MuPDFCore/MuPDF.cs

7.56 MuPDFCore.TXTCreationOptions Class Reference

Options for creating a text or structured text document.

Properties

- bool `IncludeAnnotations` = true [get, set]

If this is true, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.
- bool `InhibitSpaces` = false [get, set]

Do not add spaces between gaps in the text.
- bool `PreserveLigatures` = false [get, set]

Do not expand ligatures into constituent characters.
- bool `PreserveWhitespace` = false [get, set]

Do not convert all whitespace into space characters.
- bool `PreserveSpans` = false [get, set]

Do not merge spans on the same line.
- bool `Dehyphenate` = false [get, set]

Attempt to join up hyphenated words.
- bool `UseCIDForUnknownUnicode` = false [get, set]

Guess unicode from CID if normal mapping fails.
- bool `IncludeCharactersOutsideMediaBox` = false [get, set]

Include characters that are outside of the page's mediabox.

7.56.1 Detailed Description

Options for creating a text or structured text document.

Definition at line 934 of file [MuPDFDocument.Create.cs](#).

7.56.2 Property Documentation

7.56.2.1 Dehyphenate

```
bool MuPDFCore.TXTCreationOptions.Dehyphenate = false [get], [set]
```

Attempt to join up hyphenated words.

Definition at line 964 of file [MuPDFDocument.Create.cs](#).

7.56.2.2 IncludeAnnotations

```
bool MuPDFCore.TXTCreationOptions.IncludeAnnotations = true [get], [set]
```

If this is `true`, annotations (e.g. signatures) are included in the converted document. Otherwise, only the page contents are included.

Definition at line 939 of file [MuPDFDocument.Create.cs](#).

7.56.2.3 IncludeCharactersOutsideMediaBox

```
bool MuPDFCore.TXTCreationOptions.IncludeCharactersOutsideMediaBox = false [get], [set]
```

Include characters that are outside of the page's mediabox.

Definition at line 974 of file [MuPDFDocument.Create.cs](#).

7.56.2.4 InhibitSpaces

```
bool MuPDFCore.TXTCreationOptions.InhibitSpaces = false [get], [set]
```

Do not add spaces between gaps in the text.

Definition at line 944 of file [MuPDFDocument.Create.cs](#).

7.56.2.5 PreserveLigatures

```
bool MuPDFCore.TXTCreationOptions.PreserveLigatures = false [get], [set]
```

Do not expand ligatures into constituent characters.

Definition at line 949 of file [MuPDFDocument.Create.cs](#).

7.56.2.6 PreserveSpans

```
bool MuPDFCore.TXTCreationOptions.PreserveSpans = false [get], [set]
```

Do not merge spans on the same line.

Definition at line 959 of file [MuPDFDocument.Create.cs](#).

7.56.2.7 PreserveWhitespace

```
bool MuPDFCore.TXTCreationOptions.PreserveWhitespace = false [get], [set]
```

Do not convert all whitespace into space characters.

Definition at line 954 of file [MuPDFDocument.Create.cs](#).

7.56.2.8 UseCIDForUnknownUnicode

```
bool MuPDFCore.TXTCreationOptions.UseCIDForUnknownUnicode = false [get], [set]
```

Guess unicode from CID if normal mapping fails.

Definition at line 969 of file [MuPDFDocument.Create.cs](#).

The documentation for this class was generated from the following file:

- [MuPDFCore/MuPDFDocument.Create.cs](#)

Chapter 8

File Documentation

8.1 PDFRenderer.cs

```
00001 /*
00002 MuPDFCore.MuPDFRenderer - A control to display documents in Avalonia using MuPDFCore.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using Avalonia;
00019 using Avalonia.Animation;
00020 using Avalonia.Controls;
00021 using Avalonia.Input;
00022 using Avalonia.Layout;
00023 using Avalonia.LogicalTree;
00024 using Avalonia.Media;
00025 using Avalonia.Media.Imaging;
00026 using Avalonia.Platform;
00027 using Avalonia.Threading;
00028 using MuPDFCore.StructuredText;
00029 using System;
00030 using System.Collections.Generic;
00031 using System.IO;
00032 using System.Linq;
00033 using System.Runtime.InteropServices;
00034 using System.Text.RegularExpressions;
00035 using System.Threading;
00036 using System.Threading.Tasks;
00037
00038 namespace MuPDFCore.MuPDFRenderer
00039 {
00040 /// <summary>
00041 /// A control to render PDF documents (and other formats), potentially using multiple threads.
00042 /// </summary>
00043     public partial class PDFRenderer : Control
00044     {
00045     /// <summary>
00046     /// If this is true, the <see cref="Context"/> and <see cref="Document"/> will be disposed when this
00047     /// object is detached from the logical tree.
00048         private bool OwnsContextAndDocument = true;
00049
00050     /// <summary>
00051     /// The <see cref="MuPDFContext"/> using which the <see cref="Document"/> was created.
00052     /// </summary>
00053         protected MuPDFContext Context;
00054
00055     /// <summary>
00056     /// The <see cref="MuPDFDocument"/> from which the <see cref="Renderer"/> was created.
00057     /// </summary>
```

```

00058     protected MuPDFDocument Document;
00059
00060     /// <summary>
00061     /// The <see cref="MuPDFMultiThreadedPageRenderer"/> that renders the dynamic tiles.
00062     /// </summary>
00063     private MuPDFMultiThreadedPageRenderer Renderer;
00064
00065     /// <summary>
00066     /// The static renderisation of the page.
00067     /// </summary>
00068     private WriteableBitmap FixedCanvasBitmap;
00069
00070     /// <summary>
00071     /// The area covered by the <see cref="FixedCanvasBitmap"/>. It should be equal to the <see
00072     /// cref="PageSize"/>, but doesn't have to.
00073     /// </summary>
00074     private Rectangle FixedArea;
00075
00076     /// <summary>
00077     /// The position and size of the dynamic tiles.
00078     /// </summary>
00079     private RoundedRectangle[] DynamicImagesBounds;
00080
00081     /// <summary>
00082     /// The dynamic tiles.
00083     /// </summary>
00084     private WriteableBitmap[] DynamicBitmaps;
00085
00086     /// <summary>
00087     /// If this is true, the <see cref="DynamicBitmaps"/> have been rendered and can be drawn on screen.
00088     /// </summary>
00089     private bool AreDynamicBitmapsReady = false;
00090
00091     /// <summary>
00092     /// If this is true, the <see cref="DynamicBitmaps"/> will be rendered again immediately after the
00093     /// current rendering operation finishes.
00094     /// </summary>
00095     private bool RenderQueued = false;
00096
00097     /// <summary>
00098     /// A <see cref="Mutex"/> to synchronise rendering operations. If someone else is holding this mutex,
00099     /// you can assume that it's not safe to access the <see cref="DynamicBitmaps"/>.
00100     /// </summary>
00101     private Mutex RenderMutex;
00102
00103     /// <summary>
00104     /// A <see cref="Geometry"/> holding the icon that is displayed in the top-right corner when the <see
00105     /// cref="DynamicBitmaps"/> are not available.
00106     /// </summary>
00107     private PathGeometry RefreshingGeometry;
00108
00109     /// <summary>
00110     /// The thread that is in charge of responding to the rendering requests and either starting a new
00111     /// rendering of the <see cref="DynamicBitmaps"/>, or queueing it.
00112     /// </summary>
00113     private Thread RenderDynamicCanvasOuterThread;
00114
00115     /// <summary>
00116     /// The thread that is in charge of rendering the <see cref="DynamicBitmaps"/>.
00117     /// </summary>
00118     private Thread RenderDynamicCanvasInnerThread;
00119
00120     /// <summary>
00121     /// An <see cref="EventWaitHandle"/> that signals a request for rendering to the <see
00122     /// cref="RenderDynamicCanvasOuterThread"/>.
00123     /// </summary>
00124     private readonly EventWaitHandle RenderDynamicCanvasOuterHandle = new EventWaitHandle(false,
00125         EventResetMode.ManualReset);
00126
00127     /// <summary>
00128     /// An <see cref="EventWaitHandle"/> that signals to the <see cref="RenderDynamicCanvasOuterThread"/>
00129     /// that the <see cref="RenderDynamicCanvasInnerThread"/> has acquired the <see cref="RenderMutex"/> and
00130     /// is starting rendering.
00131     /// </summary>
00132     private readonly EventWaitHandle RenderDynamicCanvasInnerStartedHandle = new
00133         EventWaitHandle(false, EventResetMode.ManualReset);
00134
00135     /// <summary>
00136     /// An <see cref="EventWaitHandle"/> that signals to the <see cref="RenderDynamicCanvasOuterThread"/>
00137     /// and the <see cref="RenderDynamicCanvasInnerThread"/> to cease all operation because this <see

```

```
    cref="PDFRenderer"/> is being detached from the logical tree.  
00132 /// </summary>  
00133     private readonly EventWaitHandle RendererDisposedHandle = new EventWaitHandle(false,  
    EventResetMode.ManualReset);  
00134 /// <summary>  
00135 /// The current rendering resolution (in screen units) that is used by the renderer when rendering the  
    <see cref="DynamicBitmaps"/>.  
00136 /// </summary>  
00137     private readonly int[] RenderSize = new int[2];  
00138  
00139 /// <summary>  
00140 /// The area on the page that will be rendered by the renderer in the <see cref="DynamicBitmaps"/>.  
00141 /// </summary>  
00142     private Rect RenderDisplayArea;  
00143  
00144 /// <summary>  
00145 /// A lock to prevent race conditions when multiple rendering passes are queued consecutively.  
00146 /// </summary>  
00147     private readonly object RenderDisplayAreaLock = new object();  
00148  
00149 /// <summary>  
00150 /// Whether a PointerPressed event has fired.  
00151 /// </summary>  
00152     private bool IsMouseDown = false;  
00153  
00154 /// <summary>  
00155 /// The point at which the PointerPressed event fired.  
00156 /// </summary>  
00157     private Point MouseDownPoint;  
00158  
00159 /// <summary>  
00160 /// The <see cref="DisplayArea"/> when the PointerPressed event fired.  
00161 /// </summary>  
00162     private Rect MouseDownDisplayArea;  
00163  
00164 /// <summary>  
00165 /// A structured text representation of the current page, used for selection and search highlight.  
00166 /// </summary>  
00167     protected MuPDFStructuredTextPage StructuredTextPage;  
00168  
00169 /// <summary>  
00170 /// A list of <see cref="Quad"/>s that cover the selected text region.  
00171 /// </summary>  
00172     protected List<Quad> SelectionQuads;  
00173  
00174 /// <summary>  
00175 /// A list of <see cref="Quad"/>s that cover the highlighted regions.  
00176 /// </summary>  
00177     protected List<Quad> HighlightQuads;  
00178  
00179 /// <summary>  
00180 /// Defines the current mouse operation.  
00181 /// </summary>  
00182     private enum CurrentMouseOperations  
00183     {  
00184         Pan,  
00185  
00186         /// The mouse is being used to pan around the page.  
00187     /// </summary>  
00188         Pan,  
00189  
00190         /// <summary>  
00191         /// The mouse is being used to highlight text  
00192     /// </summary>  
00193         Highlight  
00194     }  
00195  
00196 /// <summary>  
00197 /// The current mouse operation.  
00198 /// </summary>  
00199     private CurrentMouseOperations CurrentMouseOperation;  
00200  
00201 /// <summary>  
00202 /// Initializes a new instance of the <see cref="PDFRenderer"/> class.  
00203 /// </summary>  
00204     public PDFRenderer()  
00205     {  
00206         this.InitializeComponent();  
00207  
00208         this.PropertyChanged += ControlPropertyChanged;  
00209         this.DetachedFromLogicalTree += ControlDetachedFromLogicalTree;  
00210         this.PointerPressed += ControlPointerPressed;  
00211         this.PointerReleased += ControlPointerReleased;  
00212         this.PointerMoved += ControlPointerMoved;  
00213         this.PointerWheelChanged += ControlPointerWheelChanged;  
00214     }  
00215
```

```

00216 /// <summary>
00217 /// Initializes inner components of the <see cref="PDFRenderer"/>.
00218 /// </summary>
00219     private void InitializeComponent()
00220     {
00221         PathFigure arrow1 = new PathFigure
00222         {
00223             StartPoint = new Point(16 * Math.Cos(Math.PI / 4), 16 * Math.Sin(Math.PI / 4))
00224         };
00225         arrow1.Segments.Add(new ArcSegment() { Point = new Point(-16, 0), IsLargeArc = false,
00226             RotationAngle = 0, Size = new Avalonia.Size(16, 16), SweepDirection = SweepDirection.Clockwise });
00226         arrow1.Segments.Add(new LineSegment() { Point = new Point(-7.2727, 0) });
00227         arrow1.Segments.Add(new LineSegment() { Point = new Point(-21.8181, -17.4545) });
00228         arrow1.Segments.Add(new LineSegment() { Point = new Point(-36.3636, 0) });
00229         arrow1.Segments.Add(new LineSegment() { Point = new Point(-27.6363, 0) });
00230         arrow1.Segments.Add(new ArcSegment() { Point = new Point(27.6363 * Math.Cos(Math.PI / 4),
00231             27.6363 * Math.Sin(Math.PI / 4)), IsLargeArc = false, RotationAngle = 0, Size = new
00232             Avalonia.Size(27.6363, 27.6363), SweepDirection = SweepDirection.CounterClockwise });
00231         arrow1.IsClosed = true;
00232
00233         PathFigure arrow2 = new PathFigure
00234         {
00235             StartPoint = new Point(16 * Math.Cos(5 * Math.PI / 4), 16 * Math.Sin(5 * Math.PI / 4))
00236         };
00237         arrow2.Segments.Add(new ArcSegment() { Point = new Point(16, 0), IsLargeArc = false,
00238             RotationAngle = 0, Size = new Avalonia.Size(16, 16), SweepDirection = SweepDirection.Clockwise });
00238         arrow2.Segments.Add(new LineSegment() { Point = new Point(7.2727, 0) });
00239         arrow2.Segments.Add(new LineSegment() { Point = new Point(21.8181, 17.4545) });
00240         arrow2.Segments.Add(new LineSegment() { Point = new Point(36.3636, 0) });
00241         arrow2.Segments.Add(new LineSegment() { Point = new Point(27.6363, 0) });
00242         arrow2.Segments.Add(new ArcSegment() { Point = new Point(27.6363 * Math.Cos(5 * Math.PI /
00243             4), 27.6363 * Math.Sin(5 * Math.PI / 4)), IsLargeArc = false, RotationAngle = 0, Size = new
00244             Avalonia.Size(27.6363, 27.6363), SweepDirection = SweepDirection.CounterClockwise });
00243         arrow2.IsClosed = true;
00244
00245         RefreshingGeometry = new PathGeometry();
00246         RefreshingGeometry.Figures.Add(arrow1);
00247         RefreshingGeometry.Figures.Add(arrow2);
00248     }
00249
00250 /// <summary>
00251 /// Set up the <see cref="PDFRenderer"/> to display a page of a <see cref="MuPDFDocument"/>.
00252 /// </summary>
00253 /// <param name="document">The <see cref="MuPDFDocument"/> to render.</param>
00254 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00254 // appropriate number of threads based on the number of processors in the computer will be used.
00254 // Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00254 // biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00254 // used.</param>
00255 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00255 // 0.</param>
00256 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00256 // at which the static renderisation of the page will be produced. If <paramref
00256 // name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00256 // cref="PDFRenderer"/>.</param>
00257 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00257 // signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00258 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00258 // null, no OCR is performed.</param>
00259     public void Initialize(MuPDFDocument document, int threadCount = 0, int pageNumber = 0, double
00259 // resolutionMultiplier = 1, bool includeAnnotations = true, TesseractLanguage ocrLanguage = null)
00260     {
00261         if (IsViewerInitialized)
00262         {
00263             ReleaseResources();
00264         }
00265
00266         OwnsContextAndDocument = false;
00267
00268         Document = document;
00269         Context = null;
00270
00271         ContinueInitialization(threadCount, pageNumber, resolutionMultiplier, includeAnnotations,
00271             ocrLanguage);
00272     }
00273
00274 /// <summary>
00275 /// Set up the <see cref="PDFRenderer"/> to display a page of a <see cref="MuPDFDocument"/>. The OCR
00275 // step is run asynchronously, in order not to block the UI thread.
00276 /// </summary>
00277 /// <param name="document">The <see cref="MuPDFDocument"/> to render.</param>
00278 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00278 // appropriate number of threads based on the number of processors in the computer will be used.
00278 // Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00278 // biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00278 // used.</param>
00279 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index

```

```
0.</param>
00280 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
at which the static renderisation of the page will be produced. If <paramref
name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
 cref="PDFRenderer"/>.</param>
00281 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00282 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
null, no OCR is performed.</param>
00283 /// <param name="ocrCancellationToken">A <see cref="CancellationToken"/> used to cancel the OCR
operation.</param>
00284 /// <param name="ocrProgress">An <see cref="IProgress<OCRProgressInfo>"/> used to report OCR
progress.</param>
00285         public async Task InitializeAsync(MuPDFDocument document, int threadCount = 0, int pageNumber
= 0, double resolutionMultiplier = 1, bool includeAnnotations = true, TesseractLanguage ocrLanguage =
null, CancellationToken ocrCancellationToken = default, IProgress<OCRProgressInfo> ocrProgress = null)
00286     {
00287         if (IsViewerInitialized)
00288         {
00289             ReleaseResources();
00290         }
00291
00292         OwnsContextAndDocument = false;
00293
00294         Document = document;
00295         Context = null;
00296
00297         await ContinueInitializationAsync(threadCount, pageNumber, resolutionMultiplier,
includeAnnotations, ocrLanguage, ocrCancellationToken, ocrProgress);
00298     }
00299
00300 /// <summary>
00301 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from
disk.
00302 /// </summary>
00303 /// <param name="fileName">The path to the document that should be opened.</param>
00304 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
appropriate number of threads based on the number of processors in the computer will be used.
Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
used.</param>
00305 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
0.</param>
00306 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
at which the static renderisation of the page will be produced. If <paramref
name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
 cref="PDFRenderer"/>.</param>
00307 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00308 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
null, no OCR is performed.</param>
00309         public void Initialize(string fileName, int threadCount = 0, int pageNumber = 0, double
resolutionMultiplier = 1, bool includeAnnotations = true, TesseractLanguage ocrLanguage = null)
00310     {
00311         if (IsViewerInitialized)
00312         {
00313             ReleaseResources();
00314         }
00315
00316         OwnsContextAndDocument = true;
00317
00318         Context = new MuPDFContext();
00319         Document = new MuPDFDocument(Context, fileName);
00320
00321         ContinueInitialization(threadCount, pageNumber, resolutionMultiplier, includeAnnotations,
ocrLanguage);
00322     }
00323
00324 /// <summary>
00325 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from
disk. The OCR step is run asynchronously, in order not to block the UI thread.
00326 /// </summary>
00327 /// <param name="fileName">The path to the document that should be opened.</param>
00328 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
appropriate number of threads based on the number of processors in the computer will be used.
Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
used.</param>
00329 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
0.</param>
00330 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
at which the static renderisation of the page will be produced. If <paramref
name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
 cref="PDFRenderer"/>.</param>
00331 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00332 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
```

```

        null, no OCR is performed.</param>
00333 /// <param name="ocrCancellationToken">A <see cref="CancellationToken"/> used to cancel the OCR
00334 /// operation.</param>
00335 /// <param name="ocrProgress">An <see cref="IPrgress<OCRProgressInfo>"> used to report OCR
00336     public async Task InitializeAsync(string fileName, int threadCount = 0, int pageNumber = 0,
00337                                         double resolutionMultiplier = 1, bool includeAnnotations = true, TesseractLanguage ocrLanguage = null,
00338                                         CancellationToken ocrCancellationToken = default, IPrgress<OCRProgressInfo> ocrProgress = null)
00339     {
00340         if (IsViewerInitialized)
00341         {
00342             ReleaseResources();
00343         }
00344         OwnsContextAndDocument = true;
00345         Context = new MuPDFContext();
00346         Document = new MuPDFDocument(Context, fileName);
00347         await ContinueInitializationAsync(threadCount, pageNumber, resolutionMultiplier,
00348                                         includeAnnotations, ocrLanguage, ocrCancellationToken, ocrProgress);
00349     }
00350 /// <summary>
00351 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from a
00352 /// <see cref="MemoryStream"/>.
00353 /// <param name="ms">The <see cref="MemoryStream"/> containing the document that should be opened.
00354 /// This can be safely disposed after this method returns.</param>
00355 /// <param name="fileType">The format of the document.</param>
00356 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00357 /// appropriate number of threads based on the number of processors in the computer will be used.
00358 /// Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00359 /// biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00360 /// used.</param>
00361 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00362 /// 0.</param>
00363 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00364 /// at which the static renderisation of the page will be produced. If <paramref
00365 /// name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00366 /// cref="PDFRenderer"/>.</param>
00367 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00368 /// signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00369 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00370 /// null, no OCR is performed.</param>
00371     public void Initialize(MemoryStream ms, InputFileType fileType, int threadCount = 0, int
00372                             pageNumber = 0, double resolutionMultiplier = 1, bool includeAnnotations = true, TesseractLanguage
00373                             ocrLanguage = null)
00374     {
00375         //Get the byte array that underlies the MemoryStream.
00376         int origin = (int)ms.Seek(0, SeekOrigin.Begin);
00377         long dataLength = ms.Length;
00378         byte[] dataBytes = ms.GetBuffer();
00379
00380         Initialize(dataBytes, fileType, origin, (int)dataLength, threadCount, pageNumber,
00381                     resolutionMultiplier, includeAnnotations, ocrLanguage);
00382     }
00383 /// <summary>
00384 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from a
00385 /// <see cref="MemoryStream"/>. The OCR step is run asynchronously, in order not to block the UI thread.
00386 /// <summary>
00387 /// <param name="ms">The <see cref="MemoryStream"/> containing the document that should be opened.
00388 /// This can be safely disposed after this method returns.</param>
00389 /// <param name="fileType">The format of the document.</param>
00390 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00391 /// appropriate number of threads based on the number of processors in the computer will be used.
00392 /// Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00393 /// biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00394 /// used.</param>
00395 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00396 /// 0.</param>
00397 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00398 /// at which the static renderisation of the page will be produced. If <paramref
00399 /// name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00400 /// cref="PDFRenderer"/>.</param>
00401 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00402 /// signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00403 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00404 /// null, no OCR is performed.</param>
00405     public async Task InitializeAsync(MemoryStream ms, InputFileType fileType, int threadCount =
00406                                         0, int pageNumber = 0, double resolutionMultiplier = 1, bool includeAnnotations = true,
00407                                         TesseractLanguage ocrLanguage = null, CancellationToken ocrCancellationToken = default,

```

```

IProgress<OCRProgressInfo> ocrProgress = null)
00383     {
00384         //Get the byte array that underlies the MemoryStream.
00385         int origin = (int)ms.Seek(0, SeekOrigin.Begin);
00386         long dataLength = ms.Length;
00387         byte[] dataBytes = ms.GetBuffer();
00388
00389         await InitializeAsync(dataBytes, fileType, origin, (int)dataLength, threadCount,
00390             pageNumber, resolutionMultiplier, includeAnnotations, ocrLanguage, ocrCancellationToken, ocrProgress);
00391     }
00392 /// <summary>
00393 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from an
00394 array of <see cref="byte"/>s.
00395 /// </summary>
00396 /// <param name="dataBytes">The bytes of the document that should be opened. The array will be copied
00397 and can be safely discarded/altered after this method returns.</param>
00398 /// <param name="fileType">The format of the document.</param>
00399 /// <param name="offset">The offset in the byte array at which the document starts.</param>
00400 /// <param name="length">The length of the document in bytes. If this is <code>< 0</code>, the whole array is
00401 used.</param>
00402 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00403 appropriate number of threads based on the number of processors in the computer will be used.
00404 Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00405 biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00406 used.</param>
00407 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00408 0.</param>
00409 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00410 at which the static renderisation of the page will be produced. If <paramref
00411 name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00412 cref="PDFRenderer"/>.</param>
00413 /// <param name="includeAnnotations">If this is <code><see langword="true"/></code>, annotations (e.g.
00414 signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00415 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00416 null, no OCR is performed.</param>
00417 public void Initialize(byte[] dataBytes, InputFileTypes fileType, int offset = 0, int length =
00418 -1, int threadCount = 0, int pageNumber = 0, double resolutionMultiplier = 1, bool includeAnnotations
00419 = true, TesseractLanguage ocrLanguage = null)
00420     {
00421         if (IsViewerInitialized)
00422         {
00423             ReleaseResources();
00424         }
00425         if (length < 0)
00426         {
00427             length = dataBytes.Length - offset;
00428         }
00429
00430         //Copy the bytes to unmanaged memory, so that we don't depend on the original array.
00431         IntPtr pointer = Marshal.AllocHGlobal(length);
00432         Marshal.Copy(dataBytes, offset, pointer, length);
00433
00434         //Wrap the pointer into a disposable container.
00435         IDisposable disposer = new DisposableIntPtr(pointer);
00436
00437         OwnsContextAndDocument = true;
00438
00439         Context = new MuPDFContext();
00440
00441         //Create a new document, passing the wrapped pointer so that it can be released when the
00442 Document is disposed.
00443         Document = new MuPDFDocument(Context, pointer, length, fileType, ref disposer);
00444
00445         ContinueInitialization(threadCount, pageNumber, resolutionMultiplier, includeAnnotations,
00446             ocrLanguage);
00447     }
00448
00449 /// <summary>
00450 /// Set up the <see cref="PDFRenderer"/> to display a page of a document that will be loaded from an
00451 array of <see cref="byte"/>s. The OCR step is run asynchronously, in order not to block the UI
00452 thread.
00453 /// </summary>
00454 /// <param name="dataBytes">The bytes of the document that should be opened. The array will be copied
00455 and can be safely discarded/altered after this method returns.</param>
00456 /// <param name="fileType">The format of the document.</param>
00457 /// <param name="offset">The offset in the byte array at which the document starts.</param>
00458 /// <param name="length">The length of the document in bytes. If this is <code>< 0</code>, the whole array is
00459 used.</param>
00460 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00461 appropriate number of threads based on the number of processors in the computer will be used.
00462 Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00463 biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00464 used.</param>
00465 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00466 0.</param>

```

```

00442 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00443 at which the static renderisation of the page will be produced. If <paramref
00444 name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00445 cref="PDFRenderer"/>.</param>
00446 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00447 signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00448 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00449 null, no OCR is performed.</param>
00450 /// <param name="ocrCancellationToken">A <see cref="CancellationToken"/> used to cancel the OCR
00451 operation.</param>
00452 /// <param name="ocrProgress">An <see cref="IProgress<OCRProgressInfo>"/> used to report OCR
00453 progress.</param>
00454 public async Task InitializeAsync(byte[] dataBytes, InputFileType fileType, int offset = 0,
00455 int length = -1, int threadCount = 0, int pageNumber = 0, double resolutionMultiplier = 1, bool
00456 includeAnnotations = true, TesseractLanguage ocrLanguage = null, CancellationToken
00457 ocrCancellationToken = default, IProgress<OCRProgressInfo> ocrProgress = null)
00458 {
00459     if (IsViewerInitialized)
00460     {
00461         ReleaseResources();
00462     }
00463     if (length < 0)
00464     {
00465         length = dataBytes.Length - offset;
00466     }
00467
00468     //Copy the bytes to unmanaged memory, so that we don't depend on the original array.
00469     IntPtr pointer = Marshal.AllocHGlobal(length);
00470     Marshal.Copy(dataBytes, offset, pointer, length);
00471
00472     //Wrap the pointer into a disposable container.
00473     IDisposable disposer = new DisposableIntPtr(pointer);
00474
00475     OwnsContextAndDocument = true;
00476
00477     Context = new MuPDFContext();
00478
00479     //Create a new document, passing the wrapped pointer so that it can be released when the
00480     Document is disposed.
00481     Document = new MuPDFDocument(Context, pointer, length, fileType, ref disposer);
00482
00483     await ContinueInitializationAsync(threadCount, pageNumber, resolutionMultiplier,
00484     includeAnnotations, ocrLanguage, ocrCancellationToken, ocrProgress);
00485 }
00486
00487 /// <summary>
00488 /// Common steps in the initialization process that will be performed regardless of how the <see
00489 cref="Document"/> was obtained.
00490 /// </summary>
00491 /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00492 appropriate number of threads based on the number of processors in the computer will be used.
00493 Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00494 biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00495 used.</param>
00496 /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00497 0.</param>
00498 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00499 at which the static renderisation of the page will be produced. If <paramref
00500 name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00501 cref="PDFRenderer"/>.</param>
00502 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00503 signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00504
00505     private void ContinueInitialization(int threadCount, int pageNumber, double
00506     resolutionMultiplier, bool includeAnnotations, TesseractLanguage ocrLanguage = null)
00507     {
00508         //Initialise threads and locking mechanics.
00509         if (RenderMutex == null)
00510         {
00511             RenderMutex = new Mutex(false);
00512
00513             this.RenderDynamicCanvasOuterThread = new Thread(() =>
00514             {
00515                 RenderDynamicCanvasOuterAction();
00516             });
00517
00518             this.RenderDynamicCanvasInnerThread = new Thread(() =>
00519             {
00520                 RenderDynamicCanvasInnerAction();
00521             });
00522
00523             RenderDynamicCanvasOuterThread.Start();
00524             RenderDynamicCanvasInnerThread.Start();
00525         }
00526     }

```

```

00505         //Choose an appropriate number of threads based on the number of processors in the
00506         //computer. We have an upper limit of 8 threads because more threads apparently caused reduced
00507         //performance due to the synchronisation overhead.
00508         if (threadCount <= 0)
00509     {
00510         threadCount = Math.Max(1, Math.Min(8, Environment.ProcessorCount - 2));
00511     }
00512
00513         //Create the structured text representation.
00514         this.StructuredTextPage = Document.GetStructuredTextPage(pageNumber, ocrLanguage,
00515         includeAnnotations);
00516
00517         //Create the multithreaded renderer.
00518         Renderer = Document.GetMultiThreadedRenderer(pageNumber, threadCount, includeAnnotations);
00519
00520         //Set up the properties of this control.
00521         RenderThreadCount = Renderer.ThreadCount;
00522         Rectangle bounds = Document.Pages[pageNumber].Bounds;
00523         PageSize = new Rect(new Point(bounds.X0, bounds.Y0), new Point(bounds.X1, bounds.Y1));
00524         pageNumber = pageNumber;
00525
00526         //Render the static canvas (which is used when the DynamicBitmaps are not available).
00527         RenderFixedCanvas(resolutionMultiplier);
00528
00529         //Initialize the dynamic canvas.
00530         InitializeDynamicCanvas();
00531
00532         //Set initial display area to include the whole page.
00533         double widthRatio = PageSize.Width / (this.Bounds.Width * resolutionMultiplier);
00534         double heightRatio = PageSize.Height / (this.Bounds.Height * resolutionMultiplier);
00535
00536         double containingWidth = Math.Max(widthRatio, heightRatio) * this.Bounds.Width *
00537             resolutionMultiplier;
00538         double containingHeight = Math.Max(widthRatio, heightRatio) * this.Bounds.Height *
00539             resolutionMultiplier;
00540
00541         SetDisplayAreaNowInternal(new Rect(new Point(-(containingWidth - FixedArea.Width) * 0.5,
00542             -(containingHeight - FixedArea.Height) * 0.5), new Avalonia.Size(containingWidth, containingHeight)));
00543         this._Zoom = this.Bounds.Width / DisplayArea.Width * 72 / 96 * (VisualRoot as
00544             ILayoutRoot).LayoutScaling;
00545
00546         //We are ready!
00547         IsViewerInitialized = true;
00548
00549         //Queue a render of the DynamicBitmaps (on another thread).
00550         RenderDynamicCanvas();
00551     }
00552
00553     /// <summary>
00554     /// Common steps in the initialization process that will be performed regardless of how the <see
00555     /// cref="Document"/> was obtained. The OCR step is run asynchronously, in order not to block the UI
00556     /// thread.
00557     /// </summary>
00558     /// <param name="threadCount">The number of threads to use in the rendering. If this is 0, an
00559     // appropriate number of threads based on the number of processors in the computer will be used.
00560     // Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the
00561     // biggest number smaller than <paramref name="threadCount"/> that satisfies this condition is
00562     // used.</param>
00563     /// <param name="pageNumber">The index of the page that should be rendered. The first page has index
00564     // 0.</param>
00565     /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00566     // at which the static renderisation of the page will be produced. If <paramref
00567     // name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00568     // cref="PDFRenderer"/>. </param>
00569     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00570     // signatures) are included in the rendering. Otherwise, only the page contents are included.</param>
00571     /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
00572     // null, no OCR is performed.</param>
00573     /// <param name="ocrCancellationToken">A <see cref="CancellationToken"/> used to cancel the OCR
00574     // operation.</param>
00575     /// <param name="ocrProgress">An <see cref="IProgress<OCRProgressInfo>"/> used to report OCR
00576     // progress.</param>
00577     private async Task ContinueInitializationAsync(int threadCount, int pageNumber, double
00578         resolutionMultiplier, bool includeAnnotations, TesseractLanguage ocrLanguage, CancellationToken
00579         ocrCancellationToken, IProgress<OCRProgressInfo> ocrProgress)
00580     {
00581         //Initialise threads and locking mechanics.
00582         if (RenderMutex == null)
00583     {
00584         RenderMutex = new Mutex(false);
00585
00586         this.RenderDynamicCanvasOuterThread = new Thread(() =>
00587         {
00588             RenderDynamicCanvasOuterAction();
00589         });
00590     }

```

```

00569         this.RenderDynamicCanvasInnerThread = new Thread(() =>
00570         {
00571             RenderDynamicCanvasInnerAction();
00572         });
00573 
00574         RenderDynamicCanvasOuterThread.Start();
00575         RenderDynamicCanvasInnerThread.Start();
00576     }
00577 
00578     //Choose an appropriate number of threads based on the number of processors in the
00579     //computer. We have an upper limit of 8 threads because more threads apparently caused reduced
00580     //performance due to the synchronisation overhead.
00581     if (threadCount <= 0)
00582     {
00583         threadCount = Math.Max(1, Math.Min(8, Environment.ProcessorCount - 2));
00584     }
00585 
00586     //Create the structured text representation.
00587     this.StructuredTextPage = await Document.GetStructuredTextPageAsync(pageNumber,
00588     ocrLanguage, includeAnnotations, StructuredTextFlags.None, ocrCancellationToken, ocrProgress);
00589 
00590     //Create the multithreaded renderer.
00591     Renderer = Document.GetMultiThreadedRenderer(pageNumber, threadCount, includeAnnotations);
00592 
00593     //Set up the properties of this control.
00594     RenderThreadCount = Renderer.ThreadCount;
00595     Rectangle bounds = Document.Pages[pageNumber].Bounds;
00596     PageSize = new Rect(new Point(bounds.X0, bounds.Y0), new Point(bounds.X1, bounds.Y1));
00597     pageNumber = pageNumber;
00598 
00599     //Render the static canvas (which is used when the DynamicBitmaps are not available).
00600     RenderFixedCanvas(resolutionMultiplier);
00601 
00602     //Initialize the dynamic canvas.
00603     InitializeDynamicCanvas();
00604 
00605     //Set initial display area to include the whole page.
00606     double widthRatio = PageSize.Width / (this.Bounds.Width * resolutionMultiplier);
00607     double heightRatio = PageSize.Height / (this.Bounds.Height * resolutionMultiplier);
00608 
00609     double containingWidth = Math.Max(widthRatio, heightRatio) * this.Bounds.Width *
00610     resolutionMultiplier;
00611     double containingHeight = Math.Max(widthRatio, heightRatio) * this.Bounds.Height *
00612     resolutionMultiplier;
00613 
00614     SetDisplayAreaNowInternal(new Rect(new Point(-(containingWidth - FixedArea.Width) * 0.5,
00615     -(containingHeight - FixedArea.Height) * 0.5), new Avalonia.Size(containingWidth, containingHeight)));
00616     this._Zoom = this.Bounds.Width / DisplayArea.Width * 72 / 96 * (VisualRoot as
00617     ILayoutRoot).LayoutScaling;
00618 
00619     //We are ready!
00620     IsViewerInitialized = true;
00621 
00622     //Queue a render of the DynamicBitmaps (on another thread).
00623     RenderDynamicCanvas();
00624 }
00625 
00626 /// <summary>
00627 /// Release resources held by this PDFRenderer. This is not an irreversible step: using one of the
00628 /// Initialize overloads after calling this method will restore functionality.
00629 /// </summary>
00630 public void ReleaseResources()
00631 {
00632     IsViewerInitialized = false;
00633     try
00634     {
00635         this.StructuredTextPage?.Dispose();
00636     }
00637     catch (LifetimeManagementException<MuPDFStructuredTextPage, MuPDFContext> e)
00638     {
00639         throw new LifetimeManagementException<MuPDFStructuredTextPage, MuPDFContext>(e, "If
00640         you are manually disposing the MuPDFContext accessed by this PDFRenderer control, please ensure that
00641         this happens after the control has been removed from the logical tree, e.g. in the Window.Closed
00642         event handler rather than Window.Closing.");
00643     }
00644     catch (LifetimeManagementException<MuPDFFont, MuPDFContext> e)
00645     {
00646         throw new LifetimeManagementException<MuPDFFont, MuPDFContext>(e, "If you are manually
00647         disposing the MuPDFContext accessed by this PDFRenderer control, please ensure that this happens after
00648         the control has been removed from the logical tree, e.g. in the Window.Closed event handler rather
00649         than Window.Closing.");
00650     }
00651 
00652     this.Renderer?.Dispose();
00653     this.StructuredTextPage = null;
00654     this.Selection = null;
00655     this.HighlightedRegions = null;

```

```

00642             if (OwnsContextAndDocument)
00643             {
00644                 this.Document?.Dispose();
00645                 this.Context?.Dispose();
00646             }
00647         }
00648     }
00649
00650 /// <summary>
00651 /// Called when the PDFRenderer is removed from the logical tree (e.g. it is removed from the window,
00652 // or the window containing it is closed). We assume that this renderer is not needed anymore. This is
00653 // irreversible!
00654 /// </summary>
00655     private void ControlDetachedFromLogicalTree(object sender, LogicalTreeAttachmentEventArgs e)
00656     {
00657         RendererDisposedHandle.Set();
00658         ReleaseResources();
00659     }
00660 /// <summary>
00661 /// Set the current display area to the specified <paramref name="value"/>, skipping all transitions.
00662 // This also skips sanity checks of the <paramref name="value"/>, since the calling methods will already
00663 // have performed them.
00664 /// <param name="value">The new display area.</param>
00665     private void SetDisplayAreaNowInternal(Rect value)
00666     {
00667         Transitions prevTransitions = this.Transitions;
00668         this.Transitions = null;
00669         SetValue(DisplayAreaProperty, value);
00670         this.Transitions = prevTransitions;
00671     }
00672 /// <summary>
00673 // Set the current display area to the specified <paramref name="value"/>, skipping all transitions.
00674 /// </summary>
00675 /// <param name="value">The new display area.</param>
00676     public void SetDisplayAreaNow(Rect value)
00677     {
00678         Transitions prevTransitions = this.Transitions;
00679         this.Transitions = null;
00680         this.DisplayArea = value;
00681         this.Transitions = prevTransitions;
00682     }
00683
00684 /// <summary>
00685 // Zoom around a point.
00686 /// </summary>
00687 /// <param name="count">Number of steps to zoom. Positive values indicate a zoom in, negative values
00688 // a zoom out.</param>
00689 /// <param name="center">The point around which to center the zoom operation. If this is null, the
00690 // center of the control is used.</param>
00691     public void ZoomStep(double count, Point? center = null)
00692     {
00693         if (center == null)
00694         {
00695             center = new Point(this.Bounds.Width * 0.5, this.Bounds.Height * 0.5);
00696         }
00697
00698         double currZoomX = FixedArea.Width / DisplayArea.Width;
00699         double currZoomY = FixedArea.Height / DisplayArea.Height;
00700
00701         currZoomX *= Math.Pow(ZoomIncrement, count);
00702         currZoomY *= Math.Pow(ZoomIncrement, count);
00703
00704         double currWidth = FixedArea.Width / currZoomX;
00705         double currHeight = FixedArea.Height / currZoomY;
00706
00707         double deltaW = currWidth - DisplayArea.Width;
00708         double deltaH = currHeight - DisplayArea.Height;
00709
00710         SetValue(DisplayAreaProperty, new Rect(new Point(DisplayArea.X - deltaW * center.Value.X /
00711                                         this.Bounds.Width, DisplayArea.Y - deltaH * center.Value.Y / this.Bounds.Height), new
00712                                         Point(DisplayArea.Right + deltaW * (1 - center.Value.X / this.Bounds.Width), DisplayArea.Bottom +
00713                                         deltaH * (1 - center.Value.Y / this.Bounds.Height))));
00714
00715     /// <summary>
00716     /// Alter the display area so that the whole page fits on screen.
00717     /// </summary>
00718     public void Contain()
00719     {
00720         //This will be sanitised by the property setter.
00721         this.DisplayArea = this.PageSize;
00722     }
00723

```

```

00720 /// <summary>
00721 /// Alter the display area so that the page covers the whole surface of the <see cref="PDFRenderer"/>
00722 /// (even though parts of the page may be outside it).
00723     public void Cover()
00724     {
00725         double widthRatio = this.PageSize.Width / (this.Bounds.Width);
00726         double heightRatio = this.PageSize.Height / (this.Bounds.Height);
00727
00728         double containingWidth = Math.Min(widthRatio, heightRatio) * this.Bounds.Width;
00729         double containingHeight = Math.Min(widthRatio, heightRatio) * this.Bounds.Height;
00730
00731         double deltaW = (containingWidth - this.PageSize.Width) * 0.5;
00732         double deltaH = (containingHeight - this.PageSize.Height) * 0.5;
00733
00734         Rect newDispArea = new Rect(new Point(this.PageSize.X - deltaW, this.PageSize.Y - deltaH),
00735             new Point(this.PageSize.Right + deltaW, this.PageSize.Bottom + deltaH));
00736         //Skip sanitation.
00737         SetValue(DisplayAreaProperty, newDispArea);
00738     }
00739
00740 /// <summary>
00741 /// Get the current rendering progress.
00742 /// </summary>
00743 /// <returns>A <see cref="RenderProgress"/> object with information about the rendering progress of
00744 /// each thread.</returns>
00745     public RenderProgress GetProgress()
00746     {
00747         return Renderer.GetProgress();
00748     }
00749 /// <summary>
00750 /// Get the currently selected text.
00751 /// </summary>
00752 /// <returns>The currently selected text.</returns>
00753     public string GetSelectedText()
00754     {
00755         return this.StructuredTextPage.GetText(this.Selection);
00756     }
00757
00758 /// <summary>
00759 /// Selects all the text in the document.
00760 /// </summary>
00761     public void SelectAll()
00762     {
00763         if (this.StructuredTextPage.Count > 0)
00764         {
00765             int maxBlock = this.StructuredTextPage.Count - 1;
00766             int maxLine = this.StructuredTextPage[maxBlock].Count - 1;
00767             int maxCharacter = this.StructuredTextPage[maxBlock][maxLine].Count - 1;
00768
00769             this.Selection = new MuPDFStructuredTextAddressSpan(new MuPDFStructuredTextAddress(0,
00770                 0, 0), new MuPDFStructuredTextAddress(maxBlock, maxLine, maxCharacter));
00771         }
00772         else
00773         {
00774             this.Selection = null;
00775         }
00776     }
00777 /// <summary>
00778 /// Highlights all matches of the specified <see cref="Regex"/> in the text and returns the number of
00779 /// matches found. Matches cannot span multiple lines.
00780 /// </summary>
00781 /// <param name="needle">The <see cref="Regex"/> to search for.</param>
00782 /// <returns>The number of matches that have been found.</returns>
00783     public int Search(Regex needle)
00784     {
00785         List<MuPDFStructuredTextAddressSpan> spans =
00786             this.StructuredTextPage.Search(needle).ToList();
00787         this.HighlightedRegions = spans;
00788         return spans.Count;
00789     }
00790 /// <summary>
00791 /// Render the <see cref="FixedCanvasBitmap"/>.
00792 /// </summary>
00793 /// <param name="resolutionMultiplier">This value can be used to increase or decrease the resolution
00794 /// at which the static renderisation of the page will be produced. If <paramref
00795 /// name="resolutionMultiplier"/> is 1, the resolution will match the size (in screen units) of the <see
00796 /// cref="PDFRenderer"/>.</param>
00797     private void RenderFixedCanvas(double resolutionMultiplier)
00798     {
00799         //Take into account DPI scaling.
00800         resolutionMultiplier *= (VisualRoot as ILayoutRoot)?.LayoutScaling ?? 1;

```

```

00798     double widthRatio = PageSize.Width / (this.Bounds.Width * resolutionMultiplier);
00799     double heightRatio = PageSize.Height / (this.Bounds.Height * resolutionMultiplier);
00800
00801     double zoom = 1 / Math.Min(widthRatio, heightRatio);
00802
00803     //Render the whole page
00804     Rectangle origin = new Rectangle(0, 0, PageSize.Width, PageSize.Height);
00805
00806     FixedArea = origin;
00807
00808     RoundedRectangle roundedOrigin = origin.Round(zoom);
00809
00810     RoundedSize targetSize = new RoundedSize(roundedOrigin.Width, roundedOrigin.Height);
00811     if (FixedCanvasBitmap == null)
00812     {
00813         FixedCanvasBitmap = new WriteableBitmap(new PixelSize(targetSize.Width,
00814             targetSize.Height), new Vector(72, 72), Avalonia.Platform.PixelFormat.Rgba8888, AlphaFormat.Unpremul);
00815     }
00816     else
00817     {
00818         if (FixedCanvasBitmap.PixelSize.Width != targetSize.Width ||
00819             FixedCanvasBitmap.PixelSize.Height != targetSize.Height)
00820         {
00821             FixedCanvasBitmap = new WriteableBitmap(new PixelSize(targetSize.Width,
00822                 targetSize.Height), new Vector(72, 72), Avalonia.Platform.PixelFormat.Rgba8888, AlphaFormat.Unpremul);
00823         }
00824
00825         //Render the page to the FixedCanvasBitmap (without marshaling).
00826         using (ILockedFramebuffer fb = FixedCanvasBitmap.Lock())
00827         {
00828             Document.Render(PageNumber, origin, zoom, PixelFormats.RGBA, fb.Address);
00829         }
00830     }
00831 /// <summary>
00832 /// Set up the <see cref="DynamicBitmaps"/> array with an appropriate number of <see
00833 /// cref="WriteableBitmap"/> of the appropriate size.
00834 /// </summary>
00835     private void InitializeDynamicCanvas()
00836     {
00837         //Take into account DPI scaling.
00838         double scale = (VisualRoot as ILayoutRoot)?.LayoutScaling ?? 1;
00839
00840         //Acquire the render mutex (we don't want anyone to touch the DynamicBitmaps while we are
00841         //resizing them!)
00842         RenderMutex.WaitOne();
00843         RoundedSize targetSize = new RoundedSize((int)Math.Ceiling(this.Bounds.Width * scale),
00844             (int)Math.Ceiling(this.Bounds.Height * scale));
00845
00846         //Split the target size into an appropriate number of tiles.
00847         RoundedRectangle[] splitSizes = targetSize.Split(RenderThreadCount);
00848
00849         DynamicImagesBounds = splitSizes;
00850
00851         if (DynamicBitmaps == null || DynamicBitmaps.Length != RenderThreadCount)
00852         {
00853             DynamicBitmaps = new WriteableBitmap[RenderThreadCount];
00854             for (int i = 0; i < splitSizes.Length; i++)
00855             {
00856                 DynamicBitmaps[i] = new WriteableBitmap(new PixelSize(splitSizes[i].Width,
00857                     splitSizes[i].Height), new Vector(72, 72), Avalonia.Platform.PixelFormat.Rgba8888,
00858                     AlphaFormat.Unpremul);
00859             }
00860         }
00861         else
00862         {
00863             for (int i = 0; i < splitSizes.Length; i++)
00864             {
00865                 if (DynamicBitmaps[i].PixelSize.Width != splitSizes[i].Width ||
00866                     DynamicBitmaps[i].PixelSize.Height != splitSizes[i].Height)
00867                 {
00868                     DynamicBitmaps[i] = new WriteableBitmap(new PixelSize(splitSizes[i].Width,
00869                         splitSizes[i].Height), new Vector(72, 72), Avalonia.Platform.PixelFormat.Rgba8888,
00870                         AlphaFormat.Unpremul);
00871                 }
00872             }
00873         }
00874
00875         //Release the render mutex.
00876         RenderMutex.ReleaseMutex();
00877     }
00878 }
00879 /// <summary>
00880 /// The outer loop that is executed by the <see cref="RenderDynamicCanvasOuterThread"/>, which is in
00881 /// charge of responding to the rendering requests and either starting a new rendering of the <see
00882 /// cref="DynamicBitmaps"/>, or queueing it.

```

```

00872 /// </summary>
00873     private void RenderDynamicCanvasOuterAction()
00874     {
00875         EventWaitHandle[] handles = new EventWaitHandle[] { RenderDynamicCanvasOuterHandle,
00876             RendererDisposedHandle };
00876
00877         while (true)
00878     {
00879             int result = EventWaitHandle.WaitAny(handles);
00880
00881             if (result == 0)
00882             {
00883                 //So that we don't lose consecutive requests.
00884                 RenderDynamicCanvasOuterHandle.Reset();
00885
00886                 //Check if the rendering is already in progress.
00887                 if (RenderMutex.WaitOne(0))
00888                 {
00889                     //Start a new rendering
00890                     AreDynamicBitmapsReady = false;
00891
00892                     //This handle will be set by the inner thread once it starts rendering.
00893                     RenderDynamicCanvasInnerStartedHandle.Reset();
00894
00895                     //Tell the inner thread to start rendering.
00896                     RenderDynamicCanvasInnerHandle.Set();
00897
00898                     //Release the mutex so that the inner thread can start rendering.
00899                     RenderMutex.ReleaseMutex();
00900
00901                     //Wait until the inner thread has acuired the mutex and started rendering.
00902                     RenderDynamicCanvasInnerStartedHandle.WaitOne();
00903                 }
00904             else
00905             {
00906                 if (!RenderQueued)
00907                 {
00908                     //Queue another rendering pass.
00909                     RenderQueued = true;
00910
00911                     //Abort the current rendering pass.
00912                     Renderer.Abort();
00913                 }
00914             }
00915         }
00916     else
00917     {
00918         //The renderer is being disposed, we need to quit!
00919         break;
00920     }
00921 }
00922 }
00923
00924 /// <summary>
00925 /// The inner loop that is executed by the <see cref="RenderDynamicCanvasInnerThread"/>, which renders
00926 /// the <see cref="DynamicBitmaps"/>.
00927 /// </summary>
00928     private void RenderDynamicCanvasInnerAction()
00929     {
00930         EventWaitHandle[] handles = new EventWaitHandle[] { RenderDynamicCanvasInnerHandle,
00931             RendererDisposedHandle };
00932
00933         bool ownsMutex = false;
00934
00935         while (true)
00936     {
00937         int result = EventWaitHandle.WaitAny(handles);
00938
00939         if (result == 0)
00940         {
00941             //So that we don't lose consecutive requests.
00942             RenderDynamicCanvasInnerHandle.Reset();
00943
00944             //Acquire the mutex only if have not acquired it yet.
00945             if (!ownsMutex)
00946             {
00947                 RenderMutex.WaitOne();
00948                 ownsMutex = true;
00949
00950                 //Signal to the outer thread that we have acquired the mutex. Even if the outer
00951                 //thread is not waiting for this signal, it will reset it before waiting for it.
00952                 RenderDynamicCanvasInnerStartedHandle.Set();
00953
00954                 //Set up the pointers to the contents of the DynamicBitmaps
00955                 IntPtr[] destinations = new IntPtr[RenderThreadCount];
00956                 ILockedFramebuffer[] fbs = new ILockedFramebuffer[RenderThreadCount];

```

```
00955             for (int i = 0; i < RenderThreadCount; i++)
00956             {
00957                 fbs[i] = DynamicBitmaps[i].Lock();
00958                 destinations[i] = fbs[i].Address;
00959             }
00960
00961             //Prevent race conditions.
00962             Rectangle target;
00963             int width;
00964             int height;
00965             lock (RenderDisplayAreaLock)
00966             {
00967                 target = new Rectangle(RenderDisplayArea.X, RenderDisplayArea.Y,
00968                 RenderDisplayArea.Right, RenderDisplayArea.Bottom);
00969                 width = RenderSize[0];
00970                 height = RenderSize[1];
00971             }
00972
00973             //Start the multithreaded rendering and wait until it finishes.
00974             Renderer.Render(new RoundedSize(width, height), target, destinations,
00975             PixelFormats.RGBA);
00976
00977             //Free the pointers.
00978             for (int i = 0; i < RenderThreadCount; i++)
00979             {
00980                 fbs[i].Dispose();
00981             }
00982
00983             //Check whether another rendering request has been queued.
00984             if (!RenderQueued)
00985             {
00986                 //No other rendering requests have been queued.
00987                 AreDynamicBitmapsReady = true;
00988
00989                 //This should always be true. Release the rendering mutex.
00990                 if (ownsMutex)
00991                 {
00992                     RenderMutex.ReleaseMutex();
00993                     ownsMutex = false;
00994                 }
00995
00996                 //Signal to the UI that a repaint is required.
00997                 Dispatcher.UIThread.InvokeAsync(() =>
00998                 {
00999                     this.InvalidateVisual();
01000                 });
01001             }
01002             else
01003             {
01004                 //Another rendering request has been queued, we can assume that whatever we
01005                 //have rendered until now is useless (maybe because the rendering has been aborted).
01006                 RenderQueued = false;
01007
01008                 //Self-signal.
01009                 RenderDynamicCanvasInnerHandle.Set();
01010             }
01011         }
01012         else
01013         {
01014             //The renderer is being disposed, we need to quit!
01015             break;
01016         }
01017     }
01018 /// <summary>
01019 /// Signal to the <see cref="RenderDynamicCanvasOuterThread"/> that a rendering has been requested.
01020 /// </summary>
01021     private void RenderDynamicCanvas()
01022     {
01023         //Take into account DPI scaling.
01024         double scale = (VisualRoot as ILayoutRoot).LayoutScaling;
01025
01026         //Set up rendering size
01027         lock (RenderDisplayAreaLock)
01028         {
01029             RenderSize[0] = (int)Math.Ceiling(this.Bounds.Width * scale);
01030             RenderSize[1] = (int)Math.Ceiling(this.Bounds.Height * scale);
01031             RenderDisplayArea = DisplayArea;
01032         }
01033
01034         //Send the signal.
01035         RenderDynamicCanvasOuterHandle.Set();
01036     }
01037
01038 /// <summary>
```

```

01039 /// Resizes the <see cref="DynamicBitmaps"/> when the size of the control changes and queues a repaint
01040 /// when the <see cref="DisplayArea"/> changes.
01041 /// <param name="sender"></param>
01042 /// <param name="e"></param>
01043     private void ControlPropertyChanged(object sender, AvaloniaPropertyChangedEventArgs e)
01044     {
01045         if (e.Property == UserControl.BoundsProperty)
01046         {
01047             if (IsViewerInitialized)
01048             {
01049                 //Resize the display area
01050                 Rect oldBounds = (Rect)e.OldValue;
01051                 Rect newBounds = (Rect)e.NewValue;
01052
01053                 double deltaW = (newBounds.Width - oldBounds.Width) / oldBounds.Width *
01054                     DisplayArea.Width;
01055                 double deltaH = (newBounds.Height - oldBounds.Height) / oldBounds.Height *
01056                     DisplayArea.Height;
01057
01058                 Rect target = new Rect(new Point(DisplayArea.X - deltaW * 0.5, DisplayArea.Y -
01059                     deltaH * 0.5), new Point(DisplayArea.Right + deltaW * 0.5, DisplayArea.Bottom + deltaH * 0.5));
01060
01061                 //Resize the DynamicBitmaps
01062                 InitializeDynamicCanvas();
01063
01064                 //Set the new DisplayArea, skipping any animation.
01065                 SetDisplayAreaNowInternal(target);
01066             }
01067         else if (e.Property == PDFRenderer.DisplayAreaProperty)
01068         {
01069             if (IsViewerInitialized)
01070             {
01071                 //Update the value of the Zoom property.
01072                 ComputeZoom();
01073
01074                 //Signal that a repaint is needed
01075                 this.InvalidateVisual();
01076
01077                 //Queue a new rendering of the DynamicBitmaps
01078                 RenderDynamicCanvas();
01079             }
01080         else if (e.Property == PDFRenderer.SelectionProperty)
01081         {
01082             if (this.StructuredTextPage != null)
01083             {
01084                 //Update the selection quads to reflect the new selection
01085                 this.SelectionQuads = this.StructuredTextPage.GetHighlightQuads(this.Selection,
01086                     false).ToList();
01087             }
01088             else
01089             {
01090                 this.SelectionQuads = null;
01091             }
01092             this.InvalidateVisual();
01093         else if (e.Property == PDFRenderer.HighlightedRegionsProperty && this.StructuredTextPage
01094             != null)
01095         {
01096             //Update the highlight quads to reflect the new highlighted regions
01097             this.HighlightQuads = new List<Quad>();
01098
01099             if (this.HighlightedRegions != null)
01100             {
01101                 foreach (MuPDFStructuredTextAddressSpan span in this.HighlightedRegions)
01102                 {
01103                     this.HighlightQuads.AddRange(this.StructuredTextPage.GetHighlightQuads(span,
01104                         false));
01105                 }
01106             }
01107         }
01108
01109 /// <summary>
01110 /// Default handler for the PointerPressed event (start panning).
01111 /// </summary>
01112 /// <param name="sender"></param>
01113 /// <param name="e"></param>
01114     private void ControlPointerPressed(object sender, PointerPressedEventArgs e)
01115     {
01116         if (this.ActivateLinks)
01117         {
01118             if (this.Document.Pages[this.PageNumber].Links?.Count > 0)

```

```
01119             {
01120                 Point point = e.GetPosition(this);
01121
01122                 MuPDFLinks links = this.Document.Pages[this.PageNumber].Links;
01123
01124                 foreach (MuPDFLink link in links)
01125                 {
01126                     if (link.IsVisible)
01127                     {
01128                         Point topLeft = new Point((link.ActiveArea.X0 - this.DisplayArea.Left) *
01129                             this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y0 - this.DisplayArea.Top) *
01130                             this.Bounds.Height / this.DisplayArea.Height);
01131                         Point bottomRight = new Point((link.ActiveArea.X1 - this.DisplayArea.Left) *
01132                             this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y1 - this.DisplayArea.Top) *
01133                             this.Bounds.Height / this.DisplayArea.Height);
01134
01135                         Rect rect = new Rect(topLeft, bottomRight);
01136                         if (rect.Contains(point))
01137                         {
01138                             LinkClicked?.Invoke(this, new
01139                                 MupdfLinkClickedEventArgs(link.Destination));
01140                         }
01141                     }
01142
01143
01144                 if (PointerEventHandlersType == PointerEventHandlers.Pan)
01145                 {
01146                     if (e.GetCurrentPoint(this).Properties.PointerUpdateKind ==
01147                         PointerUpdateKind.LeftButtonPressed)
01148                     {
01149                         IsMouseDown = true;
01150                         MouseDownPoint = e.GetPosition(this);
01151                         MouseDownDisplayArea = DisplayArea;
01152                         this.Cursor = new Cursor(StandardCursorType.SizeAll);
01153                     }
01154                     else if (PointerEventHandlersType == PointerEventHandlers.Highlight)
01155                     {
01156                         if (e.GetCurrentPoint(this).Properties.PointerUpdateKind ==
01157                             PointerUpdateKind.LeftButtonPressed)
01158                         {
01159                             Point point = e.GetPosition(this);
01160                             IsMouseDown = true;
01161                             MouseDownPoint = point;
01162                             MouseDownDisplayArea = DisplayArea;
01163
01164                             PointF pagePoint = new PointF((float)(point.X / this.Bounds.Width *
01165                             DisplayArea.Width + DisplayArea.Left), (float)(point.Y / this.Bounds.Height * DisplayArea.Height +
01166                             DisplayArea.Top));
01167
01168                             MuPDFStructuredTextAddress? address =
01169                             StructuredTextPage?.GetHitAddress(pagePoint, false);
01170
01171                             if (address != null)
01172                             {
01173                                 this.Selection = new MuPDFStructuredTextAddressSpan(address.Value, null);
01174                             }
01175                         }
01176                     }
01177                 }
01178                 else if (PointerEventHandlersType == PointerEventHandlers.PanHighlight)
01179                 {
01180                     Point point = e.GetPosition(this);
01181                     PointF pagePoint = new PointF((float)(point.X / this.Bounds.Width * DisplayArea.Width +
01182                         DisplayArea.Left), (float)(point.Y / this.Bounds.Height * DisplayArea.Height + DisplayArea.Top));
01183                     MuPDFStructuredTextAddress? address = StructuredTextPage?.GetHitAddress(pagePoint,
01184                         false);
01185
01186                     if (address == null)
01187                     {
01188                         if (e.GetCurrentPoint(this).Properties.PointerUpdateKind ==
01189                             PointerUpdateKind.LeftButtonPressed)
01190                         {
01191                             IsMouseDown = true;
01192                             MouseDownPoint = e.GetPosition(this);
01193                             MouseDownDisplayArea = DisplayArea;
01194                             this.Cursor = new Cursor(StandardCursorType.SizeAll);
01195                             CurrentMouseOperation = CurrentMouseOperations.Pan;
```

```

01193             }
01194         }
01195     }
01196     {
01197         if (e.GetCurrentPoint(this).Properties.PointerUpdateKind ==
01198             PointerUpdateKind.LeftButtonPressed)
01199         {
01200             IsMouseDown = true;
01201             MouseDownPoint = point;
01202             MouseDownDisplayArea = DisplayArea;
01203
01204             this.Selection = new MuPDFStructuredTextAddressSpan(address.Value, null);
01205             CurrentMouseOperation = CurrentMouseOperations.Highlight;
01206         }
01207     }
01208 }
01209
01210 /// <summary>
01211 /// Default handler for the PointerReleased event (finish panning).
01212 /// </summary>
01213 /// <param name="sender"></param>
01214 /// <param name="e"></param>
01215     private void ControlPointerReleased(object sender, PointerReleasedEventArgs e)
01216     {
01217         if (PointerEventHandlersType == PointerEventHandlers.Pan)
01218         {
01219             if (e.InitialPressMouseButton == MouseButton.Left)
01220             {
01221                 IsMouseDown = false;
01222                 this.Cursor = new Cursor(StandardCursorType.Arrow);
01223             }
01224         }
01225         else if (PointerEventHandlersType == PointerEventHandlers.Highlight)
01226         {
01227             if (e.InitialPressMouseButton == MouseButton.Left)
01228             {
01229                 IsMouseDown = false;
01230                 if (e.GetPosition(this).Equals(MouseDownPoint))
01231                 {
01232                     this.Selection = null;
01233                 }
01234             }
01235         }
01236         else if (PointerEventHandlersType == PointerEventHandlers.PanHighlight)
01237         {
01238             if (e.InitialPressMouseButton == MouseButton.Left)
01239             {
01240                 IsMouseDown = false;
01241                 if (CurrentMouseOperation == CurrentMouseOperations.Pan)
01242                 {
01243                     this.Cursor = new Cursor(StandardCursorType.Arrow);
01244                 }
01245                 if (e.GetPosition(this).Equals(MouseDownPoint))
01246                 {
01247                     this.Selection = null;
01248                 }
01249             }
01250         }
01251     }
01252
01253 /// <summary>
01254 /// Default handler for the PointerMoved event (pan).
01255 /// </summary>
01256 /// <param name="sender"></param>
01257 /// <param name="e"></param>
01258     private void ControlPointerMoved(object sender, PointerEventArgs e)
01259     {
01260         if (IsMouseDown)
01261         {
01262             if (PointerEventHandlersType == PointerEventHandlers.Pan || (PointerEventHandlersType
01263 == PointerEventHandlers.PanHighlight && CurrentMouseOperation == CurrentMouseOperations.Pan))
01264             {
01265                 Point point = eGetPosition(this);
01266
01267                 double deltaX = (-point.X + MouseDownPoint.X) / this.Bounds.Width *
01268                     DisplayArea.Width;
01269                 double deltaY = (-point.Y + MouseDownPoint.Y) / this.Bounds.Height *
01270                     DisplayArea.Height;
01271
01272                 Rect target = new Rect(new Point(this.MouseDownDisplayArea.X + deltaX,
01273                     this.MouseDownDisplayArea.Y + deltaY), new Point(this.MouseDownDisplayArea.Right + deltaX,
01274                     this.MouseDownDisplayArea.Bottom + deltaY));
01275
01276                 SetDisplayAreaNowInternal(target);
01277                 this.Cursor = new Cursor(StandardCursorType.SizeAll);
01278             }
01279         }
01280     }

```

```
01274             }
01275         }
01276         else if (PointerEventHandlersType == PointerEventHandlers.Highlight ||
01277             (PointerEventHandlersType == PointerEventHandlers.PanHighlight && CurrentMouseOperation ==
01278             CurrentMouseOperations.Highlight))
01279     {
01280         Point point = e.GetPosition(this);
01281
01282         PointF pagePoint = new PointF((float)(point.X / this.Bounds.Width *
01283             DisplayArea.Width + DisplayArea.Left), (float)(point.Y / this.Bounds.Height * DisplayArea.Height +
01284             DisplayArea.Top));
01285
01286         MuPDFStructuredTextAddress? address =
01287             StructuredTextPage?.GetClosestHitAddress(pagePoint, false);
01288
01289         this.Selection = new MuPDFStructuredTextAddressSpan(this.Selection.Start,
01290             address);
01291
01292         if (address != null)
01293     {
01294         this.Cursor = new Cursor(StandardCursorType.Ibeam);
01295     }
01296     else
01297     {
01298         this.Cursor = new Cursor(StandardCursorType.Arrow);
01299     }
01300
01301     else if (PointerEventHandlersType == PointerEventHandlers.Highlight ||
01302             PointerEventHandlersType == PointerEventHandlers.PanHighlight)
01303     {
01304         Point point = e.GetPosition(this);
01305
01306         PointF pagePoint = new PointF((float)(point.X / this.Bounds.Width *
01307             DisplayArea.Width + DisplayArea.Left), (float)(point.Y / this.Bounds.Height * DisplayArea.Height +
01308             DisplayArea.Top));
01309
01310         MuPDFStructuredTextAddress? address =
01311             StructuredTextPage?.GetHitAddress(pagePoint, false);
01312
01313         if (address != null)
01314     {
01315             this.Cursor = new Cursor(StandardCursorType.Ibeam);
01316         }
01317         else
01318     {
01319         this.Cursor = new Cursor(StandardCursorType.Arrow);
01320     }
01321
01322         if (this.ActivateLinks)
01323     {
01324         if (this.Document.Pages[this.PageNumber].Links?.Count > 0)
01325     {
01326             Point point = e.GetPosition(this);
01327
01328             MuPDFLinks links = this.Document.Pages[this.PageNumber].Links;
01329
01330             foreach (MuPDFLink link in links)
01331         {
01332             if (link.IsVisible)
01333         {
01334                 Point topLeft = new Point((link.ActiveArea.X0 - this.DisplayArea.Left) *
01335                     this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y0 - this.DisplayArea.Top) *
01336                     this.Bounds.Height / this.DisplayArea.Height);
01337
01338                 Point bottomRight = new Point((link.ActiveArea.X1 -
01339                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y1 -
01340                     this.DisplayArea.Top) * this.Bounds.Height / this.DisplayArea.Height);
01341
01342                 Rect rect = new Rect(topLeft, bottomRight);
01343
01344                 if (rect.Contains(point))
01345             {
01346                 this.Cursor = new Cursor(StandardCursorType.Hand);
01347                 break;
01348             }
01349         }
01350     }
01351 }
```

```

01347             }
01348         }
01349
01350     /// <summary>
01351     /// Default handler for the PointerWheelChanged event (zoom in/out).
01352     /// </summary>
01353     /// <param name="sender"></param>
01354     /// <param name="e"></param>
01355     private void ControlPointerWheelChanged(object sender, PointerWheelEventArgs e)
01356     {
01357         if (ZoomEnabled)
01358         {
01359             ZoomStep(e.Delta.Y, e.GetPosition(this));
01360         }
01361     }
01362
01363     /// <summary>
01364     /// Compute the current value of the <see cref="Zoom"/> property.
01365     /// </summary>
01366     private void ComputeZoom()
01367     {
01368         //Take into account DPI scaling.
01369         double scale = (VisualRoot as ILayoutRoot).LayoutScaling;
01370         SetAndRaise(ZoomProperty, ref _Zoom, this.Bounds.Width / DisplayArea.Width * 72 / 96 *
01371         scale);
01372
01373     /// <summary>
01374     /// Draw the rendered document.
01375     /// </summary>
01376     /// <param name="context">The drawing context on which to draw.</param>
01377     public override void Render(DrawingContext context)
01378     {
01379         //Take into account DPI scaling.
01380         double scale = (VisualRoot as ILayoutRoot)?.LayoutScaling ?? 1;
01381
01382         context.FillRectangle(Background, new Rect(this.Bounds.Size));
01383
01384         //Page boundaries (used to draw the page background).
01385         double minX = Math.Max(PageSize.Left, DisplayArea.Left);
01386         double maxX = Math.Min(PageSize.Right, DisplayArea.Right);
01387         double minY = Math.Max(PageSize.Top, DisplayArea.Top);
01388         double maxY = Math.Min(PageSize.Bottom, DisplayArea.Bottom);
01389
01390
01391         if (IsViewerInitialized)
01392         {
01393             bool renderedDynamic = false;
01394
01395             //Check if someone is holding the mutex without blocking.
01396             if (RenderMutex.WaitOne(0))
01397             {
01398                 //Check if the DynamicBitmaps are ready
01399                 if (AreDynamicBitmapsReady)
01400                 {
01401                     //Page background
01402                     context.FillRectangle(PageBackground, new Rect(new Point((minX -
01403                         DisplayArea.Left) / DisplayArea.Width * this.Bounds.Width, (minY - DisplayArea.Top) /
01404                         DisplayArea.Height * this.Bounds.Height), new Point((maxX - DisplayArea.Left) / DisplayArea.Width *
01405                         this.Bounds.Width, (maxY - DisplayArea.Top) / DisplayArea.Height * this.Bounds.Height)));
01406
01407                     //Draw the DynamicBitmaps.
01408                     for (int i = 0; i < DynamicImagesBounds.Length; i++)
01409                     {
01410                         context.DrawImage(DynamicBitmaps[i], new Rect(new Point(0, 0),
01411                             DynamicBitmaps[i].PixelSize.ToSize(1)), new Rect(DynamicImagesBounds[i].X0 / scale,
01412                             DynamicImagesBounds[i].Y0 / scale, DynamicImagesBounds[i].Width / scale, DynamicImagesBounds[i].Height /
01413                             scale));
01414
01415                     }
01416
01417                     //Signal that we don't need to draw the static image.
01418                     renderedDynamic = true;
01419
01420                 }
01421
01422                 //Release the mutex.
01423                 RenderMutex.ReleaseMutex();
01424
01425             }
01426
01427             //If the DynamicBitmaps have not been drawn, we fall back to drawing the static image
01428             //which will probably be ugly and pixelated, but better than nothing).
01429             if (!renderedDynamic)
01430             {
01431                 //Page background
01432                 context.FillRectangle(PageBackground, new Rect(new Point((minX - DisplayArea.Left) /
01433                     DisplayArea.Width * this.Bounds.Width, (minY - DisplayArea.Top) / DisplayArea.Height *
01434                     this.Bounds.Height), new Point((maxX - DisplayArea.Left) / DisplayArea.Width * this.Bounds.Width,
01435                     (maxY - DisplayArea.Top) / DisplayArea.Height * this.Bounds.Height)));
01436
01437         }
01438
01439     }
01440
01441 
```

```

01423
01424             //Top left corner of the DisplayArea in FixedCanvasBitmap coordinates.
01425             Point topLeft = new Point((DisplayArea.X - FixedArea.X0) / FixedArea.Width *
01426             FixedCanvasBitmap.PixelSize.Width, (DisplayArea.Y - FixedArea.Y0) / FixedArea.Height *
01427             FixedCanvasBitmap.PixelSize.Height);
01428             //Size of the DisplayArea in FixedCanvasBitmap coordinates.
01429             Avalonia.Size size = new Avalonia.Size(DisplayArea.Width / FixedArea.Width *
01430             FixedCanvasBitmap.PixelSize.Width, DisplayArea.Height / FixedArea.Height *
01431             FixedCanvasBitmap.PixelSize.Height);
01432             //Draw the FixedCanvasBitmap
01433             context.DrawImage(FixedCanvasBitmap, new Rect(topLeft, size), new Rect(0, 0,
01434             this.Bounds.Width, this.Bounds.Height));
01435             //Draw the icon signaling that the DynamicBitmaps are still being rendered.
01436             RefreshingGeometry.Transform = new TranslateTransform(this.Bounds.Width - 38, 32);
01437             context.DrawGeometry(new SolidColorBrush(Color.FromRgb(119, 170, 221)), null,
01438             RefreshingGeometry);
01439         }
01440         //Draw the highlight quads
01441         if (this.HighlightQuads != null && this.HighlightQuads.Count > 0)
01442         {
01443             PathGeometry highlightGeometry = new PathGeometry() { FillRule = FillRule.NonZero
01444             };
01445             for (int i = 0; i < this.HighlightQuads.Count; i++)
01446             {
01447                 Point ll = new Point((this.HighlightQuads[i].LowerLeft.X -
01448                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01449                     (this.HighlightQuads[i].LowerLeft.Y - this.DisplayArea.Top) * this.Bounds.Height /
01450                     this.DisplayArea.Height);
01451                 Point ul = new Point((this.HighlightQuads[i].UpperLeft.X -
01452                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01453                     (this.HighlightQuads[i].UpperLeft.Y - this.DisplayArea.Top) * this.Bounds.Height /
01454                     this.DisplayArea.Height);
01455                 Point ur = new Point((this.HighlightQuads[i].UpperRight.X -
01456                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01457                     (this.HighlightQuads[i].UpperRight.Y - this.DisplayArea.Top) * this.Bounds.Height /
01458                     this.DisplayArea.Height);
01459                 Point lr = new Point((this.HighlightQuads[i].LowerRight.X -
01460                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01461                     (this.HighlightQuads[i].LowerRight.Y - this.DisplayArea.Top) * this.Bounds.Height /
01462                     this.DisplayArea.Height);
01463                 PathFigure quad = new PathFigure() { StartPoint = ll, IsClosed = true,
01464                     IsFilled = true };
01465                 quad.Segments.Add(new LineSegment() { Point = ul });
01466                 quad.Segments.Add(new LineSegment() { Point = ur });
01467                 quad.Segments.Add(new LineSegment() { Point = lr });
01468                 highlightGeometry.Figures.Add(quad);
01469             }
01470             context.DrawGeometry(this.HighlightBrush, null, highlightGeometry);
01471         }
01472         //Draw the selection quads
01473         if (this.SelectionQuads != null && this.SelectionQuads.Count > 0)
01474         {
01475             PathGeometry selectionGeometry = new PathGeometry() { FillRule = FillRule.NonZero
01476             };
01477             for (int i = 0; i < this.SelectionQuads.Count; i++)
01478             {
01479                 Point ll = new Point((this.SelectionQuads[i].LowerLeft.X -
01480                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01481                     (this.SelectionQuads[i].LowerLeft.Y - this.DisplayArea.Top) * this.Bounds.Height /
01482                     this.DisplayArea.Height);
01483                 Point ul = new Point((this.SelectionQuads[i].UpperLeft.X -
01484                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01485                     (this.SelectionQuads[i].UpperLeft.Y - this.DisplayArea.Top) * this.Bounds.Height /
01486                     this.DisplayArea.Height);
01487                 Point ur = new Point((this.SelectionQuads[i].UpperRight.X -
01488                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01489                     (this.SelectionQuads[i].UpperRight.Y - this.DisplayArea.Top) * this.Bounds.Height /
01490                     this.DisplayArea.Height);
01491                 Point lr = new Point((this.SelectionQuads[i].LowerRight.X -
01492                     this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width,
01493                     (this.SelectionQuads[i].LowerRight.Y - this.DisplayArea.Top) * this.Bounds.Height /
01494                     this.DisplayArea.Height);
01495                 PathFigure quad = new PathFigure() { StartPoint = ll, IsClosed = true,
01496                     IsFilled = true };
01497                 quad.Segments.Add(new LineSegment() { Point = ul });
01498                 quad.Segments.Add(new LineSegment() { Point = ur });

```

```

01476             quad.Segments.Add(new LineSegment() { Point = lr });
01477
01478             selectionGeometry.Figures.Add(quad);
01479         }
01480
01481         context.DrawGeometry(this.SelectionBrush, null, selectionGeometry);
01482     }
01483
01484     if (this.DrawLinks)
01485     {
01486         if (this.Document.Pages[this.PageNumber].Links?.Count > 0)
01487         {
01488             MuPDFLinks links = this.Document.Pages[this.PageNumber].Links;
01489
01490             foreach (MuPDFLink link in links)
01491             {
01492                 if (link.IsVisible)
01493                 {
01494                     Point topLeft = new Point((link.ActiveArea.X0 - this.DisplayArea.Left) *
01495 * this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y0 - this.DisplayArea.Top) *
01496 this.Bounds.Height / this.DisplayArea.Height);
01497                     Point bottomRight = new Point((link.ActiveArea.X1 -
01498 this.DisplayArea.Left) * this.Bounds.Width / this.DisplayArea.Width, (link.ActiveArea.Y1 -
01499 this.DisplayArea.Top) * this.Bounds.Height / this.DisplayArea.Height);
01500
01501                     context.DrawRectangle(this.LinkBrush, this.LinkPen, new Rect(topLeft,
01502 bottomRight));
01503                 }
01504             }
01505         }
01506     }

```

8.2 PDFRenderer.Properties.cs

```

00001 /*
00002 MuPDFCore.MuPDFRenderer - A control to display documents in Avalonia using MuPDFCore.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using Avalonia;
00019 using Avalonia.Controls;
00020 using Avalonia.Media;
00021 using MuPDFCore.StructuredText;
00022 using System;
00023 using System.Collections.Generic;
00024
00025 namespace MuPDFCore.MuPDFRenderer
00026 {
00027     public partial class PDFRenderer : Control
00028     {
00029     /// <summary>
00030     /// Defines the <see cref="RenderThreadCount"/> property.
00031     /// </summary>
00032     public static readonly DirectProperty<PDFRenderer, int> RenderThreadCountProperty =
00033     AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(RenderThreadCount), o =>
00034     o.RenderThreadCount);
00035     /// <summary>
00036     /// Backing field for the <see cref="RenderThreadCount"/> property.
00037     /// </summary>
00038     private int _RenderThreadCount;
00039     /// <summary>
00040     /// Exposes the number of threads that the current instance is using to render the document.
00041     /// Read-only.
00042     /// </summary>
00043     public int RenderThreadCount
00044     {
00045         get

```

```
00043         {
00044             return _RenderThreadCount;
00045         }
00046     }
00047     private set
00048     {
00049         SetAndRaise(RenderThreadCountProperty, ref _RenderThreadCount, value);
00050     }
00051 }
00052
00053 /// <summary>
00054 /// Defines the <see cref="PageNumber"/> property.
00055 /// </summary>
00056     public static readonly DirectProperty<PDFRenderer, int> PageNumberProperty =
AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(PageNumber), o => o.PageNumber);
00057 /// <summary>
00058 /// Backing field for the <see cref="PageNumber"/> property.
00059 /// </summary>
00060     private int _PageNumber;
00061 /// <summary>
00062 /// Exposes the number of the page that the current instance is rendering. Read-only.
00063 /// </summary>
00064     public int PageNumber
00065     {
00066         get
00067         {
00068             return _PageNumber;
00069         }
00070     }
00071     private set
00072     {
00073         SetAndRaise(PageNumberProperty, ref _PageNumber, value);
00074     }
00075 }
00076
00077 /// <summary>
00078 /// Defines the <see cref="IsViewerInitialized"/> property.
00079 /// </summary>
00080     public static readonly DirectProperty<PDFRenderer, bool> IsViewerInitializedProperty =
AvaloniaProperty.RegisterDirect<PDFRenderer, bool>(nameof(IsViewerInitialized), o =>
o.IsViewerInitialized);
00081 /// <summary>
00082 /// Backing field for the <see cref="IsViewerInitialized"/> property.
00083 /// </summary>
00084     private bool _IsViewerInitialized = false;
00085 /// <summary>
00086 /// Whether the current instance has been initialised with a document to render or not. Read-only.
00087 /// </summary>
00088     public bool IsViewerInitialized
00089     {
00090         get
00091         {
00092             return _IsViewerInitialized;
00093         }
00094     }
00095     private set
00096     {
00097         SetAndRaise(IsViewerInitializedProperty, ref _IsViewerInitialized, value);
00098     }
00099 }
00100
00101 /// <summary>
00102 /// Defines the <see cref="PageSize"/> property.
00103 /// </summary>
00104     public static readonly DirectProperty<PDFRenderer, Rect> PageSizeProperty =
AvaloniaProperty.RegisterDirect<PDFRenderer, Rect>(nameof(PageSize), o => o.PageSize);
00105 /// <summary>
00106 /// Backing field for the <see cref="PageSize"/> property.
00107 /// </summary>
00108     private Rect _PageSize;
00109 /// <summary>
00110 /// Exposes the size of the page that is drawn by the current instance (in page units).
00111 /// </summary>
00112     public Rect PageSize
00113     {
00114         get
00115         {
00116             return _PageSize;
00117         }
00118     }
00119     private set
00120     {
00121         SetAndRaise(PageSizeProperty, ref _PageSize, value);
00122     }
00123 }
00124
00125 /// <summary>
```

```

00126 /// Defines the <see cref="DisplayArea"/> property.
00127 /// </summary>
00128     public static readonly StyledProperty<Rect> DisplayAreaProperty =
    AvaloniaProperty.Register<PDFRenderer, Rect>(nameof(DisplayArea));
00129 /// <summary>
00130 /// The region of the page (in page units) that is currently displayed by the current instance. This
00131 /// always has the same aspect ratio of the bounds of this control.
00132 /// When this is set, the value is sanitised so that the smallest rectangle with the correct aspect
00133 /// ratio containing the requested value is chosen.
00134     public Rect DisplayArea
00135     {
00136         get
00137         {
00138             return GetValue(DisplayAreaProperty);
00139         }
00140         set
00141         {
00142             double widthRatio = value.Width / (this.Bounds.Width);
00143             double heightRatio = value.Height / (this.Bounds.Height);
00144
00145             double containingWidth = Math.Max(widthRatio, heightRatio) * this.Bounds.Width;
00146             double containingHeight = Math.Max(widthRatio, heightRatio) * this.Bounds.Height;
00147
00148             double deltaW = (containingWidth - value.Width) * 0.5;
00149             double deltaH = (containingHeight - value.Height) * 0.5;
00150
00151             Rect newDispArea = new Rect(new Point(value.X - deltaW, value.Y - deltaH), new
Point(value.Right + deltaW, value.Bottom + deltaH));
00152
00153             SetValue(DisplayAreaProperty, newDispArea);
00154         }
00155     }
00156
00157 /// <summary>
00158 /// Defines the <see cref="ZoomIncrement"/> property.
00159 /// </summary>
00160     public static readonly StyledProperty<double> ZoomIncrementProperty =
    AvaloniaProperty.Register<PDFRenderer, double>(nameof(ZoomIncrement), Math.Pow(2, 1.0 / 3.0),
defaultBindingMode: Avalonia.Data.BindingMode.TwoWay);
00161 /// <summary>
00162 /// Determines by how much the scale will be increased/decreased by the <see cref="ZoomStep(double,
Point?)"> method. Set this to a value smaller than 1 to invert the zoom in/out direction.
00163 /// </summary>
00164     public double ZoomIncrement
00165     {
00166         get { return GetValue(ZoomIncrementProperty); }
00167
00168         set
00169         {
00170             if (value <= 0)
00171             {
00172                 throw new ArgumentOutOfRangeException(nameof(ZoomIncrement), value, "The
ZoomIncrement must be greater than 0!");
00173             }
00174
00175             SetValue(ZoomIncrementProperty, value);
00176         }
00177     }
00178
00179 /// <summary>
00180 /// Defines the <see cref="Background"/> property.
00181 /// </summary>
00182     public static readonly StyledProperty<IBrush> BackgroundProperty =
    AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(Background));
00183 /// <summary>
00184 /// The background colour of the control.
00185 /// </summary>
00186     public IBrush Background
00187     {
00188         get { return GetValue(BackgroundProperty); }
00189         set { SetValue(BackgroundProperty, value); }
00190     }
00191
00192 /// <summary>
00193 /// Defines the <see cref="PageBackground"/> property.
00194 /// </summary>
00195     public static readonly StyledProperty<IBrush> PageBackgroundProperty =
    AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(PageBackground));
00196 /// <summary>
00197 /// The background colour to use for the page drawn by the control.
00198 /// </summary>
00199     public IBrush PageBackground
00200     {
00201         get { return GetValue(PageBackgroundProperty); }
00202         set { SetValue(PageBackgroundProperty, value); }

```

```
00203         }
00204
00205     /// <summary>
00206     /// Defines the <see cref="Zoom"/> property.
00207     /// </summary>
00208     public static readonly DirectProperty<PDFRenderer, double> ZoomProperty =
00209         AvaloniaProperty.RegisterDirect<PDFRenderer, double>(nameof(Zoom), o => o.Zoom, (o, v) => o.Zoom = v,
00210         defaultBindingMode: Avalonia.Data.BindingMode.TwoWay);
00211     /// <summary>
00212     private double _Zoom;
00213     /// <summary>
00214     /// The current zoom level. Setting this will change the <see cref="DisplayArea"/> appropriately,
00215     /// zooming around the center of the <see cref="DisplayArea"/>.
00216     public double Zoom
00217     {
00218         get
00219         {
00220             return _Zoom;
00221         }
00222
00223         set
00224         {
00225             double actualZoom = value / _Zoom;
00226
00227             double currZoomX = FixedArea.Width / DisplayArea.Width * actualZoom;
00228             double currZoomY = FixedArea.Height / DisplayArea.Height * actualZoom;
00229
00230             double currWidth = FixedArea.Width / currZoomX;
00231             double currHeight = FixedArea.Height / currZoomY;
00232
00233
00234             Point pos = new Point(this.Bounds.Width * 0.5, this.Bounds.Height * 0.5);
00235
00236             double deltaW = currWidth - DisplayArea.Width;
00237             double deltaH = currHeight - DisplayArea.Height;
00238
00239             SetValue(DisplayAreaProperty, new Rect(new Point(DisplayArea.X - deltaW * pos.X /
00240                 this.Bounds.Width, DisplayArea.Y - deltaH * pos.Y / this.Bounds.Height), new Point(DisplayArea.Right +
00241                 deltaW * (1 - pos.X / this.Bounds.Width), DisplayArea.Bottom + deltaH * (1 - pos.Y /
00242                 this.Bounds.Height)));
00243         }
00244     }
00245     /// <summary>
00246     /// Identifies the action to perform on pointer events.
00247     /// </summary>
00248     public enum PointerEventHandlers
00249     {
00250         /// <summary>
00251         /// Pointer events will be used to pan around the page.
00252         Pan,
00253
00254         /// <summary>
00255         /// Pointer events will be used to highlight text.
00256         Highlight,
00257
00258         /// <summary>
00259         /// Pointer events will be used to pan around the page or to highlight text, depending on where they
00260         /// start.
00261         PanHighlight,
00262
00263         /// <summary>
00264         /// Pointer events will be ignored. If you use this value, you will have to implement your own way to
00265         /// pan around the document by changing the <see cref="DisplayArea"/> or to select text.
00266         Custom
00267     }
00268
00269     /// <summary>
00270     /// Defines the <see cref="PointerEventHandlersType"/> property.
00271     /// </summary>
00272     public static readonly StyledProperty<PointerEventHandlers> PointerEventHandlerTypeProperty =
00273         AvaloniaProperty.Register<PDFRenderer, PointerEventHandlers>(nameof(PointerEventHandlersType),
00274         PointerEventHandlers.PanHighlight);
00275     /// <summary>
00276     /// Determines which default handlers for pointer events (which are used for panning around the page)
00277     /// should be enabled. If this is <see cref="PointerEventHandlers.Custom"/>, you will have to implement
00278     /// your own way to pan around the document by changing the <see cref="DisplayArea"/>.
00279     /// </summary>
00280     public PointerEventHandlers PointerEventHandlersType
00281     {
```

```

00278         get { return GetValue(PointerEventHandlerTypeProperty); }
00279         set { SetValue(PointerEventHandlerTypeProperty, value); }
00280     }
00281
00282 /// <summary>
00283 /// Defines the <see cref="ZoomEnabled"/> property.
00284 /// </summary>
00285     public static readonly StyledProperty<bool> ZoomEnabledProperty =
AvaloniaProperty.Register<PDFRenderer, bool>(nameof(ZoomEnabled), true);
00286 /// <summary>
00287 /// Whether the default handlers for pointer wheel events (which are used for zooming in/out) should
be enabled. If this is false, you will have to implement your own way to zoom by changing the <see
cref="DisplayArea"/>.
00288 /// </summary>
00289     public bool ZoomEnabled
00290     {
00291         get { return GetValue(ZoomEnabledProperty); }
00292         set { SetValue(ZoomEnabledProperty, value); }
00293     }
00294
00295 /// <summary>
00296 /// Defines the <see cref="Selection"/> property.
00297 /// </summary>
00298     public static readonly StyledProperty<MuPDFStructuredTextAddressSpan> SelectionProperty =
AvaloniaProperty.Register<PDFRenderer, MuPDFStructuredTextAddressSpan>(nameof(Selection), null);
00299 /// <summary>
00300 /// The start and end of the currently selected text.
00301 /// </summary>
00302     public MuPDFStructuredTextAddressSpan Selection
00303     {
00304         get { return GetValue(SelectionProperty); }
00305         set { SetValue(SelectionProperty, value); }
00306     }
00307
00308 /// <summary>
00309 /// Defines the <see cref="SelectionBrush"/> property.
00310 /// </summary>
00311     public static readonly StyledProperty<IBrush> SelectionBrushProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(SelectionBrush), new
SolidColorBrush(Color.FromArgb(96, 86, 180, 233)));
00312 /// <summary>
00313 /// The colour used to highlight the <see cref="Selection"/>.
00314 /// </summary>
00315     public IBrush SelectionBrush
00316     {
00317         get { return GetValue(SelectionBrushProperty); }
00318         set { SetValue(SelectionBrushProperty, value); }
00319     }
00320
00321 /// <summary>
00322 /// Defines the <see cref="HighlightedRegions"/> property.
00323 /// </summary>
00324     public static readonly StyledProperty<IEnumerable<MuPDFStructuredTextAddressSpan>>
HighlightedRegionsProperty = AvaloniaProperty.Register<PDFRenderer,
IEnumerable<MuPDFStructuredTextAddressSpan>>(nameof(HighlightedRegions), null);
00325 /// <summary>
00326 /// A collection of highlighted regions, e.g. as a result of a text search.
00327 /// </summary>
00328     public IEnumerable<MuPDFStructuredTextAddressSpan> HighlightedRegions
00329     {
00330         get { return GetValue(HighlightedRegionsProperty); }
00331         set { SetValue(HighlightedRegionsProperty, value); }
00332     }
00333
00334 /// <summary>
00335 /// Defines the <see cref="HighlightBrush"/> property.
00336 /// </summary>
00337     public static readonly StyledProperty<IBrush> HighlightBrushProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(HighlightBrush), new
SolidColorBrush(Color.FromArgb(96, 230, 159, 0)));
00338 /// <summary>
00339 /// The colour used to highlight the <see cref="HighlightedRegions"/>.
00340 /// </summary>
00341     public IBrush HighlightBrush
00342     {
00343         get { return GetValue(HighlightBrushProperty); }
00344         set { SetValue(HighlightBrushProperty, value); }
00345     }
00346
00347 /// <summary>
00348 /// Defines the <see cref="LinkBrush"/> property.
00349 /// </summary>
00350     public static readonly StyledProperty<IBrush> LinkBrushProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(LinkBrush), null);
00351 /// <summary>
00352 /// The colour used to highlight links in the document.
00353 /// </summary>

```

```

00354     public IBrush LinkBrush
00355     {
00356         get { return GetValue(LinkBrushProperty); }
00357         set { SetValue(LinkBrushProperty, value); }
00358     }
00359
00360 /// <summary>
00361 /// Defines the <see cref="LinkPen"/> property.
00362 /// </summary>
00363     public static readonly StyledProperty<IPen> LinkPenProperty =
00364     AvaloniaProperty.Register<PDFRenderer, IPen>(nameof(LinkPen), new Pen(Color.FromRgb(0, 0,
00365     255).ToInt32()));
00366 /// <summary>
00367     /// The colour used to highlight links in the document.
00368 /// </summary>
00369     public IPen LinkPen
00370     {
00371         get { return GetValue(LinkPenProperty); }
00372         set { SetValue(LinkPenProperty, value); }
00373     }
00374 /// <summary>
00375     /// Defines the <see cref="DrawLinks"/> property.
00376 /// </summary>
00377     public static readonly StyledProperty<bool> DrawLinksProperty =
00378     AvaloniaProperty.Register<PDFRenderer, bool>(nameof(DrawLinks), true);
00379 /// <summary>
00380     /// Determines whether links are highlighted on the document.
00381     public bool DrawLinks
00382     {
00383         get { return GetValue(DrawLinksProperty); }
00384         set { SetValue(DrawLinksProperty, value); }
00385     }
00386 /// <summary>
00387     /// Defines the <see cref="ActivateLinks"/> property.
00388 /// </summary>
00389     public static readonly StyledProperty<bool> ActivateLinksProperty =
00390     AvaloniaProperty.Register<PDFRenderer, bool>(nameof(ActivateLinks), true);
00391 /// <summary>
00392     /// Determines whether links on the document can be clicked.
00393     public bool ActivateLinks
00394     {
00395         get { return GetValue(ActivateLinksProperty); }
00396         set { SetValue(ActivateLinksProperty, value); }
00397     }
00398
00399 /// <summary>
00400     /// Fired when the user clicks on a link within the document.
00401 /// </summary>
00402     public event EventHandler<MuPDFLinkClickedEventArgs> LinkClicked;
00403 }
00404
00405 /// <summary>
00406     /// <see cref="EventArgs"/> for the <see cref="PDFRenderer.LinkClicked"/> event.
00407 /// </summary>
00408     public class MuPDFLinkClickedEventArgs : EventArgs
00409     {
00410     /// <summary>
00411     /// The link destination.
00412     /// </summary>
00413     public MuPDFLinkDestination LinkDestination { get; }
00414
00415     /// <summary>
00416     /// Create a new <see cref="MuPDFLinkClickedEventArgs"/> instance.
00417     /// </summary>
00418     /// <param name="linkDestination">The link destination.</param>
00419     public MuPDFLinkClickedEventArgs(MuPDFLinkDestination linkDestination)
00420     {
00421         LinkDestination = linkDestination;
00422     }
00423 }
00424 }
```

8.3 RectTransition.cs

```

00001 /*
00002 MuPDFCore.MuPDFRenderer - A control to display documents in Avalonia using MuPDFCore.
00003 Copyright (C) 2020-2023 Giorgio Bianchini, University of Bristol
00004
00005 This program is free software: you can redistribute it and/or modify
```

```

00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
0010 but WITHOUT ANY WARRANTY; without even the implied warranty of
0011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
0012 GNU Affero General Public License for more details.
0013
0014 You should have received a copy of the GNU Affero General Public License
0015 along with this program. If not, see <http://www.gnu.org/licenses/>
0016 */
0017
0018 namespace Avalonia.Animation
0019 {
0020 /// <summary>
0021 /// Transition class that handles <see cref="AvaloniaProperty"/> with <see cref="Rect"/> types.
0022 /// </summary>
0023 public class RectTransition : InterpolatingTransitionBase<Rect>
0024 {
0025 /// <inheritdoc/>
0026     protected override Rect Interpolate(double f, Rect oldValue, Rect newValue)
0027     {
0028         return new Rect((newValue.X - oldValue.X) * f + oldValue.X,
0029                         (newValue.Y - oldValue.Y) * f + oldValue.Y,
0030                         (newValue.Width - oldValue.Width) * f + oldValue.Width,
0031                         (newValue.Height - oldValue.Height) * f + oldValue.Height);
0032     }
0033 }
0034 }
```

8.4 MuPDF.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
0010 but WITHOUT ANY WARRANTY; without even the implied warranty of
0011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
0012 GNU Affero General Public License for more details.
0013
0014 You should have received a copy of the GNU Affero General Public License
0015 along with this program. If not, see <http://www.gnu.org/licenses/>
0016 */
0017
0018 using MuPDFCore.StructuredText;
0019 using System;
0020 using System.IO;
0021 using System.IO.Pipes;
0022 using System.Runtime.InteropServices;
0023 using System.Text;
0024 using System.Threading.Tasks;
0025
0026 [assembly: System.Runtime.CompilerServices.InternalsVisibleTo("Tests,
0027     PublicKey=0024000048000094000000602000002400005253413100400001000100d18d076ff369e4fb7295f51bbfedc5974e626236cec58
0028
0029 /// <summary>
0030 /// Exit codes returned by native methods describing various errors that can occur.
0031 /// </summary>
0032 public enum ExitCodes
0033 {
0034     /// <summary>
0035     /// An error occurred while creating the context object.
0036     /// </summary>
0037     ERR_CANNOT_CREATE_CONTEXT = 129,
0038
0039     /// <summary>
0040     /// An error occurred while registering the default document handlers with the context.
0041     /// </summary>
0042     ERR_CANNOT_REGISTER_HANDLERS = 130,
0043
0044     /// <summary>
0045     /// An error occurred while opening a file.
0046     /// </summary>
0047     ERR_CANNOT_OPEN_FILE = 131,
0048
0049     /// <summary>
0050     /// An error occurred while determining the total number of pages in the document.
0051 }
```

```
00051 /// </summary>
00052     ERR_CANNOT_COUNT_PAGES = 132,
00053
00054 /// <summary>
00055 /// An error occurred while rendering the page.
00056 /// </summary>
00057     ERR_CANNOT_RENDER = 134,
00058
00059 /// <summary>
00060 /// An error occurred while opening the stream.
00061 /// </summary>
00062     ERR_CANNOT_OPEN_STREAM = 135,
00063
00064 /// <summary>
00065 /// An error occurred while loading the page.
00066 /// </summary>
00067     ERR_CANNOT_LOAD_PAGE = 136,
00068
00069 /// <summary>
00070 /// An error occurred while computing the page bounds.
00071 /// </summary>
00072     ERR_CANNOT_COMPUTE_BOUNDS = 137,
00073
00074 /// <summary>
00075 /// An error occurred while initialising the mutexes for the lock mechanism.
00076 /// </summary>
00077     ERR_CANNOT_INIT_MUTEX = 138,
00078
00079 /// <summary>
00080 /// An error occurred while cloning the context.
00081 /// </summary>
00082     ERR_CANNOT_CLONE_CONTEXT = 139,
00083
00084 /// <summary>
00085 /// An error occurred while saving the page to a raster image file.
00086 /// </summary>
00087     ERR_CANNOT_SAVE = 140,
00088
00089 /// <summary>
00090 /// An error occurred while creating the output buffer.
00091 /// </summary>
00092     ERR_CANNOT_CREATE_BUFFER = 141,
00093
00094 /// <summary>
00095 /// An error occurred while creating the document writer.
00096 /// </summary>
00097     ERR_CANNOT_CREATE_WRITER = 142,
00098
00099 /// <summary>
00100 /// An error occurred while finalising the document file.
00101 /// </summary>
00102     ERR_CANNOT_CLOSE_DOCUMENT = 143,
00103
00104 /// <summary>
00105 /// An error occurred while creating an empty structured text page.
00106 /// </summary>
00107     ERR_CANNOT_CREATE_PAGE = 144,
00108
00109 /// <summary>
00110 /// An error occurred while populating the structured text page.
00111 /// </summary>
00112     ERR_CANNOT_POPULATE_PAGE = 145,
00113
00114 /// <summary>
00115 /// An error occurred while gathering image metadata.
00116 /// </summary>
00117     ERR_IMAGE_METADATA = 146,
00118
00119 /// <summary>
00120 /// An error occurred while retrieving colour space information.
00121 /// </summary>
00122     ERR_COLORSPACE_METADATA = 147,
00123
00124 /// <summary>
00125 /// An error occurred while retrieving font metadata.
00126 /// </summary>
00127     ERR_FONT_METADATA = 148,
00128
00129 /// <summary>
00130 /// The document pointer cannot be converted to a PDF pointer.
00131 /// </summary>
00132     ERR_CANNOT_CONVERT_TO_PDF = 149,
00133
00134 /// <summary>
00135 /// No error occurred. All is well.
00136 /// </summary>
00137     EXIT_SUCCESS = 0,
```

```
00138
00139 /// <summary>
00140 /// Unknown error.
00141 /// </summary>
00142     UNKNOWN_ERROR = -1
00143 }
00144
00145 /// <summary>
00146 /// File types supported in input by the library.
00147 /// </summary>
00148     public enum InputFileTypes
00149 {
00150     /// <summary>
00151     /// Portable Document Format.
00152     /// </summary>
00153         PDF = 0,
00154
00155     /// <summary>
00156     /// XML Paper Specification document.
00157     /// </summary>
00158         XPS = 1,
00159
00160     /// <summary>
00161     /// Comic book archive file (ZIP archive containing page scans).
00162     /// </summary>
00163         CBZ = 2,
00164
00165     /// <summary>
00166     /// Portable Network Graphics format.
00167     /// </summary>
00168         PNG = 3,
00169
00170     /// <summary>
00171     /// Joint Photographic Experts Group image.
00172     /// </summary>
00173         JPEG = 4,
00174
00175     /// <summary>
00176     /// Bitmap image.
00177     /// </summary>
00178         BMP = 5,
00179
00180     /// <summary>
00181     /// Graphics Interchange Format.
00182     /// </summary>
00183         GIF = 6,
00184
00185     /// <summary>
00186     /// Tagged Image File Format.
00187     /// </summary>
00188         TIFF = 7,
00189
00190     /// <summary>
00191     /// Portable aNyMap graphics format.
00192     /// </summary>
00193         PNM = 8,
00194
00195     /// <summary>
00196     /// Portable Arbitrary Map graphics format.
00197     /// </summary>
00198         PAM = 9,
00199
00200     /// <summary>
00201     /// Electronic PUBLICATION document.
00202     /// </summary>
00203         EPUB = 10,
00204
00205     /// <summary>
00206     /// FictionBook document.
00207     /// </summary>
00208         FB2 = 11,
00209
00210     /// <summary>
00211     /// Mobipocket e-book document.
00212     /// </summary>
00213         MOBI = 12,
00214
00215     /// <summary>
00216     /// HTML document.
00217     /// </summary>
00218         HTML = 13,
00219
00220     /// <summary>
00221     /// Text document.
00222     /// </summary>
00223         TXT = 14,
00224
```

```
00225 /// <summary>
00226 /// Microsoft Word document (only text is extracted).
00227 /// </summary>
00228     DOCX = 15,
00229
00230 /// <summary>
00231 /// Microsoft Powerpoint document (only text is extracted).
00232 /// </summary>
00233     PPTX = 16,
00234
00235 /// <summary>
00236 /// Microsoft Excel document (only text is extracted).
00237 /// </summary>
00238     XLSX = 17,
00239 }
00240
00241 /// <summary>
00242 /// Raster image file types supported in output by the library.
00243 /// </summary>
00244     public enum RasterOutputFileTypes
00245 {
00246 /// <summary>
00247 /// Portable AnyMap graphics format.
00248 /// </summary>
00249     PNM = 0,
00250
00251 /// <summary>
00252 /// Portable Arbitrary Map graphics format.
00253 /// </summary>
00254     PAM = 1,
00255
00256 /// <summary>
00257 /// Portable Network Graphics format.
00258 /// </summary>
00259     PNG = 2,
00260
00261 /// <summary>
00262 /// PhotoShop Document format.
00263 /// </summary>
00264     PSD = 3,
00265
00266 /// <summary>
00267 /// Joint Photographic Experts Group format, with quality level 90.
00268 /// </summary>
00269     JPEG = 4
00270 };
00271
00272 /// <summary>
00273 /// Document file types supported in output by the library.
00274 /// </summary>
00275     public enum DocumentOutputFileTypes
00276 {
00277 /// <summary>
00278 /// Portable Document Format.
00279 /// </summary>
00280     PDF = 0,
00281
00282 /// <summary>
00283 /// Scalable Vector Graphics.
00284 /// </summary>
00285     SVG = 1,
00286
00287 /// <summary>
00288 /// Comic book archive format.
00289 /// </summary>
00290     CBZ = 2,
00291
00292 /// <summary>
00293 /// HTML format.
00294 /// </summary>
00295     HTML = 5,
00296
00297 /// <summary>
00298 /// XHTML format.
00299 /// </summary>
00300     XHTML = 6,
00301
00302 /// <summary>
00303 /// Text format.
00304 /// </summary>
00305     TXT = 7,
00306
00307 /// <summary>
00308 /// Structured text XML format.
00309 /// </summary>
00310     StructuredText = 8,
00311 };
```

```
00312
00313 /// <summary>
00314 /// Pixel formats supported by the library.
00315 /// </summary>
00316     public enum PixelFormats
00317     {
00318     /// <summary>
00319     /// 24bpp RGB format.
00320     /// </summary>
00321         RGB = 0,
00322
00323     /// <summary>
00324     /// 32bpp RGBA format.
00325     /// </summary>
00326         RGBA = 1,
00327
00328     /// <summary>
00329     /// 24bpp BGR format.
00330     /// </summary>
00331         BGR = 2,
00332
00333     /// <summary>
00334     /// 32bpp BGRA format.
00335     /// </summary>
00336         BGRA = 3
00337     }
00338
00339 /// <summary>
00340 /// Possible document encryption states.
00341 /// </summary>
00342     public enum EncryptionState
00343     {
00344     /// <summary>
00345     /// The document is not encrypted.
00346     /// </summary>
00347         Unencrypted = 0,
00348
00349     /// <summary>
00350     /// The document is encrypted and a user password is necessary to render it.
00351     /// </summary>
00352         Encrypted = 1,
00353
00354     /// <summary>
00355     /// The document is encrypted and the correct user password has been supplied.
00356     /// </summary>
00357         Unlocked = 2
00358     }
00359
00360 /// <summary>
00361 /// Possible document restriction states.
00362 /// </summary>
00363     public enum RestrictionState
00364     {
00365     /// <summary>
00366     /// The document does not have any restrictions associated to it.
00367     /// </summary>
00368         Unrestricted = 0,
00369
00370     /// <summary>
00371     /// Some restrictions apply to the document. An owner password is required to remove these
00372     /// restrictions.
00373     /// </summary>
00374         Restricted = 1,
00375
00376     /// <summary>
00377     /// The document had some restrictions and the correct owner password has been supplied.
00378     /// </summary>
00379         Unlocked = 2
00380
00381     /// <summary>
00382     /// Document restrictions.
00383     /// </summary>
00384     public enum DocumentRestrictions
00385     {
00386     /// <summary>
00387     /// No operation is restricted.
00388     /// </summary>
00389         None = 0,
00390
00391     /// <summary>
00392     /// Printing the document is restricted.
00393     /// </summary>
00394         Print = 1,
00395
00396     /// <summary>
00397     /// Copying the document is restricted.
```

```
00398 /// </summary>
00399     Copy = 2,
00400
00401 /// <summary>
00402 /// Editing the document is restricted.
00403 /// </summary>
00404     Edit = 4,
00405
00406 /// <summary>
00407 /// Annotating the document is restricted.
00408 /// </summary>
00409     Annotate = 8
00410 }
00411
00412 /// <summary>
00413 /// Password types.
00414 /// </summary>
00415     public enum PasswordTypes
00416 {
00417 /// <summary>
00418 /// No password.
00419 /// </summary>
00420     None = 0,
00421
00422 /// <summary>
00423 /// The password corresponds to the user password.
00424 /// </summary>
00425     User = 1,
00426
00427 /// <summary>
00428 /// The password corresponds to the owner password.
00429 /// </summary>
00430     Owner = 2
00431 }
00432
00433 /// <summary>
00434 /// Types of bounding boxes.
00435 /// </summary>
00436     public enum BoxType
00437 {
00438 /// <summary>
00439 /// Media box.
00440 /// </summary>
00441     MediaBox,
00442
00443 /// <summary>
00444 /// Crop box.
00445 /// </summary>
00446     CropBox,
00447
00448 /// <summary>
00449 /// Bleed box.
00450 /// </summary>
00451     BleedBox,
00452
00453 /// <summary>
00454 /// Trim box.
00455 /// </summary>
00456     TrimBox,
00457
00458 /// <summary>
00459 /// Art box.
00460 /// </summary>
00461     ArtBox,
00462
00463 /// <summary>
00464 /// Unknown box type.
00465 /// </summary>
00466     UnknownBox
00467 }
00468
00469 /// <summary>
00470 /// A struct to hold information about the current rendering process and to abort rendering as needed.
00471 /// </summary>
00472     [StructLayout(LayoutKind.Sequential)]
00473     internal struct Cookie
00474 {
00475         public int abort;
00476         public int progress;
00477         public ulong progress_max;
00478         public int errors;
00479         public int incomplete;
00480     }
00481
00482 /// <summary>
00483 /// Holds a summary of the progress of the current rendering operation.
00484 /// </summary>
```

```

00485     public class RenderProgress
00486     {
00487     /// <summary>
00488     /// Holds the progress of a single thread.
00489     /// </summary>
00490         public struct ThreadRenderProgress
00491     {
00492     /// <summary>
00493     /// The current progress.
00494     /// </summary>
00495         public int Progress;
00496
00497     /// <summary>
00498     /// The maximum progress. If this is 0, this value could not be determined (yet).
00499     /// </summary>
00500         public long MaxProgress;
00501
00502         internal ThreadRenderProgress(int progress, ulong maxProgress)
00503     {
00504         this.Progress = progress;
00505         this.MaxProgress = (long)maxProgress;
00506     }
00507 }
00508
00509     /// <summary>
00510     /// Contains the progress of all the threads used in rendering the document.
00511     /// </summary>
00512         public ThreadRenderProgress[] ThreadRenderProgresses { get; private set; }
00513
00514         internal RenderProgress(ThreadRenderProgress[] threadRenderProgresses)
00515     {
00516         ThreadRenderProgresses = threadRenderProgresses;
00517     }
00518 }
00519
00520     /// <summary>
00521     /// An <see cref="IDisposable"/> wrapper around an <see cref="IntPtr"/> that frees the allocated
00522     /// memory when it is disposed.
00523         public class DisposableIntPtr : IDisposable
00524     {
00525     /// <summary>
00526     /// The pointer to the unmanaged memory.
00527     /// </summary>
00528         private readonly IntPtr InternalPointer;
00529
00530     /// <summary>
00531     /// The number of bytes that have been allocated, for adding memory pressure.
00532     /// </summary>
00533         private readonly long BytesAllocated = -1;
00534
00535     /// <summary>
00536     /// Create a new DisposableIntPtr.
00537     /// </summary>
00538     /// <param name="pointer">The pointer that should be freed upon disposing of this object.</param>
00539         public DisposableIntPtr(IntPtr pointer)
00540     {
00541         this.InternalPointer = pointer;
00542     }
00543
00544     /// <summary>
00545     /// Create a new DisposableIntPtr, adding memory pressure to the GC to account for the allocation of
00546     /// unmanaged memory.
00547     /// <param name="pointer">The pointer that should be freed upon disposing of this object.</param>
00548     /// <param name="bytesAllocated">The number of bytes that have been allocated, for adding memory
00549     /// pressure.</param>
00550         public DisposableIntPtr(IntPtr pointer, long bytesAllocated)
00551     {
00552         this.InternalPointer = pointer;
00553         this.BytesAllocated = bytesAllocated;
00554
00555         if (BytesAllocated > 0)
00556     {
00557         GC.AddMemoryPressure(bytesAllocated);
00558     }
00559
00560         private bool disposedValue;
00561
00562     /// <inheritDoc/>
00563         protected virtual void Dispose(bool disposing)
00564     {
00565         if (!disposedValue)
00566     {
00567             Marshal.FreeHGlobal(InternalPointer);
00568

```

```
00569         if (BytesAllocated > 0)
00570         {
00571             GC.RemoveMemoryPressure(BytesAllocated);
00572         }
00573         disposedValue = true;
00574     }
00575 }
00577
00578 ///<inheritdoc/>
00579 ~DisposableIntPtr()
00580 {
00581     Dispose(disposing: false);
00582 }
00583
00584 ///<inheritdoc/>
00585 public void Dispose()
00586 {
00587     Dispose(disposing: true);
00588     GC.SuppressFinalize(this);
00589 }
00590 }
00591
00592 /// <summary>
00593 /// The exception that is thrown when a MuPDF operation fails.
00594 /// </summary>
00595 public class MuPDFException : Exception
00596 {
00597 /// <summary>
00598 /// The <see cref="ExitCodes"/> returned by the native function.
00599 /// </summary>
00600     public readonly ExitCodes ErrorCode;
00601
00602     internal MuPDFException(string message, ExitCodes errorCode) : base(message)
00603     {
00604         this.ErrorCode = errorCode;
00605     }
00606 }
00607
00608 /// <summary>
00609 /// The exception that is thrown when an attempt is made to render an encrypted document without
00610 /// supplying the required password.
00611 /// </summary>
00612 public class DocumentLockedException : Exception
00613 {
00614     internal DocumentLockedException(string message) : base(message) { }
00615 }
00616
00617 /// <summary>
00618 /// The exception that is thrown when the lifetime of disposable objects is not managed properly.
00619 /// </summary>
00620 public class LifetimeManagementException<T1, T2> : Exception
00621 {
00622     /// <inheritdoc/>
00623     public override string Message { get; }
00624
00625 /// <summary>
00626 /// The object whose disposal caused the exception.
00627     public T1 Disposing { get; }
00628
00629 /// <summary>
00630 /// The object that had already been disposed.
00631 /// </summary>
00632     public T2 Owner { get; }
00633
00634     private IntPtr DisposingPtr { get; }
00635     private IntPtr OwnerPtr { get; }
00636
00637     internal LifetimeManagementException(T1 disposing, T2 owner, IntPtr disposingPtr, IntPtr
00638     ownerPtr) : base()
00639     {
00640         this.Message = "The current " + typeof(T1).Name + " instance (" +
00641             disposingPtr.ToString("X") + ") was disposed after its owner " + typeof(T2).Name + " instance (" +
00642             ownerPtr.ToString("X") + ") had already been disposed.\n" +
00643             "This may happen if the " + typeof(T1).Name + " instance has not been properly
00644             disposed. Please ensure that all instances of classes implementing IDisposable are properly disposed
00645             in the correct order (e.g., by wrapping them in using statements).";
00646
00647     this.Disposing = disposing;
00648     this.Owner = owner;
00649 }
00650
00651 /// <summary>
00652 /// Create a new <see cref="LifetimeManagementException{T1, T2}"/> by appending the specified message
00653 /// to the original exception message.
00654 /// </summary>
```

```

00649 /// <param name="baseException">The original <see cref="LifetimeManagementException{T1,
T2}" />.</param>
00650 /// <param name="appendedMessage">The message to be appended to the original exception
00651     message.</param>
00652     public LifetimeManagementException(LifetimeManagementException<T1, T2> baseException, string
00653     appendedMessage) : this(baseException.Disposing, baseException.Owner, baseException.DisposingPtr,
00654     baseException.OwnerPtr)
00655     {
00656         this.Message += "\n" + appendedMessage;
00657     }
00658 /// <summary>
00659 /// A class to simplify passing a string to the MuPDF C library with the correct encoding.
00660     internal class UTF8EncodedString : IDisposable
00661     {
00662         private bool disposedValue;
00663
00664     /// <summary>
00665     /// The address of the bytes encoding the string in unmanaged memory.
00666     /// </summary>
00667     public IntPtr Address { get; }
00668
00669     /// <summary>
00670     /// Create a null-terminated, UTF-8 encoded string in unmanaged memory.
00671     /// </summary>
00672     /// <param name="text"></param>
00673     public UTF8EncodedString(string text)
00674     {
00675         byte[] data = System.Text.Encoding.UTF8.GetBytes(text);
00676
00677         IntPtr dataHolder = Marshal.AllocHGlobal(data.Length + 1);
00678         Marshal.Copy(data, 0, dataHolder, data.Length);
00679         Marshal.WriteByte(dataHolder, data.Length, 0);
00680
00681         this.Address = dataHolder;
00682     }
00683
00684     protected virtual void Dispose(bool disposing)
00685     {
00686         if (!disposedValue)
00687         {
00688             Marshal.FreeHGlobal(Address);
00689             disposedValue = true;
00690         }
00691     }
00692
00693     ~UTF8EncodedString()
00694     {
00695         Dispose(disposing: false);
00696     }
00697
00698     public void Dispose()
00699     {
00700         Dispose(disposing: true);
00701         GC.SuppressFinalize(this);
00702     }
00703 }
00704
00705 /// <summary>
00706 /// EventArgs for the <see cref="MuPDF.StandardOutputMessage"/> and <see
00707 /// cref="MuPDF.StandardErrorMessage"/> events.
00708     public class MessageEventArgs : EventArgs
00709     {
00710     /// <summary>
00711     /// The message that has been logged.
00712     /// </summary>
00713     public string Message { get; }
00714
00715     /// <summary>
00716     /// Create a new <see cref="MessageEventArgs"/> instance.
00717     /// </summary>
00718     /// <param name="message">The message that has been logged.</param>
00719     public MessageEventArgs(string message)
00720     {
00721         this.Message = message;
00722     }
00723 }
00724
00725     /// <summary>
00726     /// Contains static methods to perform setup operations.
00727     /// </summary>
00728     public static class MuPDF
00729     {
00730         private static int StdOutFD = -1;

```

```
00731     private static int StdErrFD = -1;
00732
00733     private static TextWriter ConsoleOut;
00734     private static TextWriter ConsoleErr;
00735
00736     private static ConsoleColor DefaultForeground;
00737     private static ConsoleColor DefaultBackground;
00738
00739     private static string PipeName;
00740     private static bool CleanupRegistered = false;
00741     private static readonly object CleanupLock = new object();
00742
00743 /// <summary>
00744 /// This event is invoked when <see cref="RedirectOutput"/> has been called and the native MuPDF
00745 /// library writes to the standard output stream.
00746 /// </summary>
00747     public static event EventHandler<MessageEventArgs> StandardOutputMessage;
00748
00749 /// <summary>
00750 /// This event is invoked when <see cref="RedirectOutput"/> has been called and the native MuPDF
00751 /// library writes to the standard error stream.
00752 /// </summary>
00753 /// <summary>
00754 /// Redirects output messages from the native MuPDF library to the <see cref="StandardOutputMessage"/>
00755 /// and <see cref="StandardErrorMessage"/> events. Note that this has side-effects.
00756 /// <returns>A <see cref="Task"/> that finishes when the output streams have been
00757 /// redirected.</returns>
00758     public static async Task RedirectOutput()
00759     {
00760         if
00761             (
00762                 System.Runtime.InteropServices.RuntimeInformation.IsOSPlatform(System.Runtime.InteropServices.OSPlatform.Windows))
00763             {
00764                 await RedirectOutputWindows();
00765             }
00766         else
00767             {
00768                 await RedirectOutputUnix();
00769             }
00770
00771         if (!CleanupRegistered)
00772             {
00773                 CleanupRegistered = true;
00774                 AppDomain.CurrentDomain.ProcessExit += (s, e) =>
00775                 {
00776                     ResetOutput();
00777                 };
00778             }
00779
00780         const int UnixMaxPipeLength = 107;
00781
00782         private static async Task RedirectOutputUnix()
00783         {
00784             if (StdOutFD < 0 && StdErrFD < 0)
00785             {
00786                 string tempPath = Path.GetTempPath();
00787
00788                 string pipeName = "MuPDFCore-" + Guid.NewGuid().ToString();
00789
00790                 pipeName = pipeName.Substring(0, Math.Min(pipeName.Length, UnixMaxPipeLength -
00791                     tempPath.Length - 4));
00792                 pipeName = Path.Combine(tempPath, pipeName);
00793
00794                 PipeName = pipeName;
00795
00796                 Task redirectOutputTask = System.Threading.Tasks.Task.Run(() =>
00797                 {
00798                     NativeMethods.RedirectOutput(out StdOutFD, out StdErrFD, pipeName + "-out",
00799                     pipeName + "-err");
00800
00801                     // Start stdout pipe (this is actually a socket)
00802                     _ = Task.Run(() =>
00803                     {
00804                         using (NamedPipeClientStream client = new NamedPipeClientStream(pipeName +
00805                             "-out"))
00806                         {
00807                             while (true)
00808                             {
00809                                 try
00810                                 {
00811                                     client.Connect(100);
00812                                     break;
00813                                 }
00814                             }
00815                         }
00816                     });
00817
00818                     // Start stderr pipe (this is actually a socket)
00819                     _ = Task.Run(() =>
00820                     {
00821                         using (NamedPipeClientStream client = new NamedPipeClientStream(pipeName +
00822                             "-err"))
00823                         {
00824                             while (true)
00825                             {
00826                                 try
00827                                 {
00828                                     client.Connect(100);
00829                                     break;
00830                                 }
00831                             }
00832                         }
00833                     });
00834
00835                     // Wait for both tasks to complete
00836                     await Task.WhenAll(redirectOutputTask, _);
00837
00838                     // Reset output streams
00839                     ResetOutput();
00840
00841                     // Set cleanup registered
00842                     CleanupRegistered = true;
00843
00844                     // Set cleanup lock
00845                     lock (CleanupLock)
00846                     {
00847                         CleanupRegistered = true;
00848                     }
00849
00850                     // Set cleanup registered
00851                     CleanupRegistered = true;
00852
00853                     // Set cleanup lock
00854                     lock (CleanupLock)
00855                     {
00856                         CleanupRegistered = true;
00857                     }
00858
00859                     // Set cleanup registered
00860                     CleanupRegistered = true;
00861
00862                     // Set cleanup lock
00863                     lock (CleanupLock)
00864                     {
00865                         CleanupRegistered = true;
00866                     }
00867
00868                     // Set cleanup registered
00869                     CleanupRegistered = true;
00870
00871                     // Set cleanup lock
00872                     lock (CleanupLock)
00873                     {
00874                         CleanupRegistered = true;
00875                     }
00876
00877                     // Set cleanup registered
00878                     CleanupRegistered = true;
00879
00880                     // Set cleanup lock
00881                     lock (CleanupLock)
00882                     {
00883                         CleanupRegistered = true;
00884                     }
00885
00886                     // Set cleanup registered
00887                     CleanupRegistered = true;
00888
00889                     // Set cleanup lock
00890                     lock (CleanupLock)
00891                     {
00892                         CleanupRegistered = true;
00893                     }
00894
00895                     // Set cleanup registered
00896                     CleanupRegistered = true;
00897
00898                     // Set cleanup lock
00899                     lock (CleanupLock)
00900                     {
00901                         CleanupRegistered = true;
00902                     }
00903
00904                     // Set cleanup registered
00905                     CleanupRegistered = true;
00906
00907                     // Set cleanup lock
00908                     lock (CleanupLock)
00909                     {
00910                         CleanupRegistered = true;
00911                     }
00912
00913                     // Set cleanup registered
00914                     CleanupRegistered = true;
00915
00916                     // Set cleanup lock
00917                     lock (CleanupLock)
00918                     {
00919                         CleanupRegistered = true;
00920                     }
00921
00922                     // Set cleanup registered
00923                     CleanupRegistered = true;
00924
00925                     // Set cleanup lock
00926                     lock (CleanupLock)
00927                     {
00928                         CleanupRegistered = true;
00929                     }
00930
00931                     // Set cleanup registered
00932                     CleanupRegistered = true;
00933
00934                     // Set cleanup lock
00935                     lock (CleanupLock)
00936                     {
00937                         CleanupRegistered = true;
00938                     }
00939
00940                     // Set cleanup registered
00941                     CleanupRegistered = true;
00942
00943                     // Set cleanup lock
00944                     lock (CleanupLock)
00945                     {
00946                         CleanupRegistered = true;
00947                     }
00948
00949                     // Set cleanup registered
00950                     CleanupRegistered = true;
00951
00952                     // Set cleanup lock
00953                     lock (CleanupLock)
00954                     {
00955                         CleanupRegistered = true;
00956                     }
00957
00958                     // Set cleanup registered
00959                     CleanupRegistered = true;
00960
00961                     // Set cleanup lock
00962                     lock (CleanupLock)
00963                     {
00964                         CleanupRegistered = true;
00965                     }
00966
00967                     // Set cleanup registered
00968                     CleanupRegistered = true;
00969
00970                     // Set cleanup lock
00971                     lock (CleanupLock)
00972                     {
00973                         CleanupRegistered = true;
00974                     }
00975
00976                     // Set cleanup registered
00977                     CleanupRegistered = true;
00978
00979                     // Set cleanup lock
00980                     lock (CleanupLock)
00981                     {
00982                         CleanupRegistered = true;
00983                     }
00984
00985                     // Set cleanup registered
00986                     CleanupRegistered = true;
00987
00988                     // Set cleanup lock
00989                     lock (CleanupLock)
00990                     {
00991                         CleanupRegistered = true;
00992                     }
00993
00994                     // Set cleanup registered
00995                     CleanupRegistered = true;
00996
00997                     // Set cleanup lock
00998                     lock (CleanupLock)
00999                     {
01000                         CleanupRegistered = true;
01001                     }
01002
01003                     // Set cleanup registered
01004                     CleanupRegistered = true;
01005
01006                     // Set cleanup lock
01007                     lock (CleanupLock)
01008                     {
01009                         CleanupRegistered = true;
01010                     }
01011
01012                     // Set cleanup registered
01013                     CleanupRegistered = true;
01014
01015                     // Set cleanup lock
01016                     lock (CleanupLock)
01017                     {
01018                         CleanupRegistered = true;
01019                     }
01020
01021                     // Set cleanup registered
01022                     CleanupRegistered = true;
01023
01024                     // Set cleanup lock
01025                     lock (CleanupLock)
01026                     {
01027                         CleanupRegistered = true;
01028                     }
01029
01030                     // Set cleanup registered
01031                     CleanupRegistered = true;
01032
01033                     // Set cleanup lock
01034                     lock (CleanupLock)
01035                     {
01036                         CleanupRegistered = true;
01037                     }
01038
01039                     // Set cleanup registered
01040                     CleanupRegistered = true;
01041
01042                     // Set cleanup lock
01043                     lock (CleanupLock)
01044                     {
01045                         CleanupRegistered = true;
01046                     }
01047
01048                     // Set cleanup registered
01049                     CleanupRegistered = true;
01050
01051                     // Set cleanup lock
01052                     lock (CleanupLock)
01053                     {
01054                         CleanupRegistered = true;
01055                     }
01056
01057                     // Set cleanup registered
01058                     CleanupRegistered = true;
01059
01060                     // Set cleanup lock
01061                     lock (CleanupLock)
01062                     {
01063                         CleanupRegistered = true;
01064                     }
01065
01066                     // Set cleanup registered
01067                     CleanupRegistered = true;
01068
01069                     // Set cleanup lock
01070                     lock (CleanupLock)
01071                     {
01072                         CleanupRegistered = true;
01073                     }
01074
01075                     // Set cleanup registered
01076                     CleanupRegistered = true;
01077
01078                     // Set cleanup lock
01079                     lock (CleanupLock)
01080                     {
01081                         CleanupRegistered = true;
01082                     }
01083
01084                     // Set cleanup registered
01085                     CleanupRegistered = true;
01086
01087                     // Set cleanup lock
01088                     lock (CleanupLock)
01089                     {
01090                         CleanupRegistered = true;
01091                     }
01092
01093                     // Set cleanup registered
01094                     CleanupRegistered = true;
01095
01096                     // Set cleanup lock
01097                     lock (CleanupLock)
01098                     {
01099                         CleanupRegistered = true;
01100                     }
01101
01102                     // Set cleanup registered
01103                     CleanupRegistered = true;
01104
01105                     // Set cleanup lock
01106                     lock (CleanupLock)
01107                     {
01108                         CleanupRegistered = true;
01109                     }
01110
01111                     // Set cleanup registered
01112                     CleanupRegistered = true;
01113
01114                     // Set cleanup lock
01115                     lock (CleanupLock)
01116                     {
01117                         CleanupRegistered = true;
01118                     }
01119
01120                     // Set cleanup registered
01121                     CleanupRegistered = true;
01122
01123                     // Set cleanup lock
01124                     lock (CleanupLock)
01125                     {
01126                         CleanupRegistered = true;
01127                     }
01128
01129                     // Set cleanup registered
01130                     CleanupRegistered = true;
01131
01132                     // Set cleanup lock
01133                     lock (CleanupLock)
01134                     {
01135                         CleanupRegistered = true;
01136                     }
01137
01138                     // Set cleanup registered
01139                     CleanupRegistered = true;
01140
01141                     // Set cleanup lock
01142                     lock (CleanupLock)
01143                     {
01144                         CleanupRegistered = true;
01145                     }
01146
01147                     // Set cleanup registered
01148                     CleanupRegistered = true;
01149
01150                     // Set cleanup lock
01151                     lock (CleanupLock)
01152                     {
01153                         CleanupRegistered = true;
01154                     }
01155
01156                     // Set cleanup registered
01157                     CleanupRegistered = true;
01158
01159                     // Set cleanup lock
01160                     lock (CleanupLock)
01161                     {
01162                         CleanupRegistered = true;
01163                     }
01164
01165                     // Set cleanup registered
01166                     CleanupRegistered = true;
01167
01168                     // Set cleanup lock
01169                     lock (CleanupLock)
01170                     {
01171                         CleanupRegistered = true;
01172                     }
01173
01174                     // Set cleanup registered
01175                     CleanupRegistered = true;
01176
01177                     // Set cleanup lock
01178                     lock (CleanupLock)
01179                     {
01180                         CleanupRegistered = true;
01181                     }
01182
01183                     // Set cleanup registered
01184                     CleanupRegistered = true;
01185
01186                     // Set cleanup lock
01187                     lock (CleanupLock)
01188                     {
01189                         CleanupRegistered = true;
01190                     }
01191
01192                     // Set cleanup registered
01193                     CleanupRegistered = true;
01194
01195                     // Set cleanup lock
01196                     lock (CleanupLock)
01197                     {
01198                         CleanupRegistered = true;
01199                     }
01200
01201                     // Set cleanup registered
01202                     CleanupRegistered = true;
01203
01204                     // Set cleanup lock
01205                     lock (CleanupLock)
01206                     {
01207                         CleanupRegistered = true;
01208                     }
01209
01210                     // Set cleanup registered
01211                     CleanupRegistered = true;
01212
01213                     // Set cleanup lock
01214                     lock (CleanupLock)
01215                     {
01216                         CleanupRegistered = true;
01217                     }
01218
01219                     // Set cleanup registered
01220                     CleanupRegistered = true;
01221
01222                     // Set cleanup lock
01223                     lock (CleanupLock)
01224                     {
01225                         CleanupRegistered = true;
01226                     }
01227
01228                     // Set cleanup registered
01229                     CleanupRegistered = true;
01230
01231                     // Set cleanup lock
01232                     lock (CleanupLock)
01233                     {
01234                         CleanupRegistered = true;
01235                     }
01236
01237                     // Set cleanup registered
01238                     CleanupRegistered = true;
01239
01240                     // Set cleanup lock
01241                     lock (CleanupLock)
01242                     {
01243                         CleanupRegistered = true;
01244                     }
01245
01246                     // Set cleanup registered
01247                     CleanupRegistered = true;
01248
01249                     // Set cleanup lock
01250                     lock (CleanupLock)
01251                     {
01252                         CleanupRegistered = true;
01253                     }
01254
01255                     // Set cleanup registered
01256                     CleanupRegistered = true;
01257
01258                     // Set cleanup lock
01259                     lock (CleanupLock)
01260                     {
01261                         CleanupRegistered = true;
01262                     }
01263
01264                     // Set cleanup registered
01265                     CleanupRegistered = true;
01266
01267                     // Set cleanup lock
01268                     lock (CleanupLock)
01269                     {
01270                         CleanupRegistered = true;
01271                     }
01272
01273                     // Set cleanup registered
01274                     CleanupRegistered = true;
01275
01276                     // Set cleanup lock
01277                     lock (CleanupLock)
01278                     {
01279                         CleanupRegistered = true;
01280                     }
01281
01282                     // Set cleanup registered
01283                     CleanupRegistered = true;
01284
01285                     // Set cleanup lock
01286                     lock (CleanupLock)
01287                     {
01288                         CleanupRegistered = true;
01289                     }
01290
01291                     // Set cleanup registered
01292                     CleanupRegistered = true;
01293
01294                     // Set cleanup lock
01295                     lock (CleanupLock)
01296                     {
01297                         CleanupRegistered = true;
01298                     }
01299
01300                     // Set cleanup registered
01301                     CleanupRegistered = true;
01302
01303                     // Set cleanup lock
01304                     lock (CleanupLock)
01305                     {
01306                         CleanupRegistered = true;
01307                     }
01308
01309                     // Set cleanup registered
01310                     CleanupRegistered = true;
01311
01312                     // Set cleanup lock
01313                     lock (CleanupLock)
01314                     {
01315                         CleanupRegistered = true;
01316                     }
01317
01318                     // Set cleanup registered
01319                     CleanupRegistered = true;
01320
01321                     // Set cleanup lock
01322                     lock (CleanupLock)
01323                     {
01324                         CleanupRegistered = true;
01325                     }
01326
01327                     // Set cleanup registered
01328                     CleanupRegistered = true;
01329
01330                     // Set cleanup lock
01331                     lock (CleanupLock)
01332                     {
01333                         CleanupRegistered = true;
01334                     }
01335
01336                     // Set cleanup registered
01337                     CleanupRegistered = true;
01338
01339                     // Set cleanup lock
01340                     lock (CleanupLock)
01341                     {
01342                         CleanupRegistered = true;
01343                     }
01344
01345                     // Set cleanup registered
01346                     CleanupRegistered = true;
01347
01348                     // Set cleanup lock
01349                     lock (CleanupLock)
01350                     {
01351                         CleanupRegistered = true;
01352                     }
01353
01354                     // Set cleanup registered
01355                     CleanupRegistered = true;
01356
01357                     // Set cleanup lock
01358                     lock (CleanupLock)
01359                     {
01360                         CleanupRegistered = true;
01361                     }
01362
01363                     // Set cleanup registered
01364                     CleanupRegistered = true;
01365
01366                     // Set cleanup lock
01367                     lock (CleanupLock)
01368                     {
01369                         CleanupRegistered = true;
01370                     }
01371
01372                     // Set cleanup registered
01373                     CleanupRegistered = true;
01374
01375                     // Set cleanup lock
01376                     lock (CleanupLock)
01377                     {
01378                         CleanupRegistered = true;
01379                     }
01380
01381                     // Set cleanup registered
01382                     CleanupRegistered = true;
01383
01384                     // Set cleanup lock
01385                     lock (CleanupLock)
01386                     {
01387                         CleanupRegistered = true;
01388                     }
01389
01390                     // Set cleanup registered
01391                     CleanupRegistered = true;
01392
01393                     // Set cleanup lock
01394                     lock (CleanupLock)
01395                     {
01396                         CleanupRegistered = true;
01397                     }
01398
01399                     // Set cleanup registered
01400                     CleanupRegistered = true;
01401
01402                     // Set cleanup lock
01403                     lock (CleanupLock)
01404                     {
01405                         CleanupRegistered = true;
01406                     }
01407
01408                     // Set cleanup registered
01409                     CleanupRegistered = true;
01410
01411                     // Set cleanup lock
01412                     lock (CleanupLock)
01413                     {
01414                         CleanupRegistered = true;
01415                     }
01416
01417                     // Set cleanup registered
01418                     CleanupRegistered = true;
01419
01420                     // Set cleanup lock
01421                     lock (CleanupLock)
01422                     {
01423                         CleanupRegistered = true;
01424                     }
01425
01426                     // Set cleanup registered
01427                     CleanupRegistered = true;
01428
01429                     // Set cleanup lock
01430                     lock (CleanupLock)
01431                     {
01432                         CleanupRegistered = true;
01433                     }
01434
01435                     // Set cleanup registered
01436                     CleanupRegistered = true;
01437
01438                     // Set cleanup lock
01439                     lock (CleanupLock)
01440                     {
01441                         CleanupRegistered = true;
01442                     }
01443
01444                     // Set cleanup registered
01445                     CleanupRegistered = true;
01446
01447                     // Set cleanup lock
01448                     lock (CleanupLock)
01449                     {
01450                         CleanupRegistered = true;
01451                     }
01452
01453                     // Set cleanup registered
01454                     CleanupRegistered = true;
01455
01456                     // Set cleanup lock
01457                     lock (CleanupLock)
01458                     {
01459                         CleanupRegistered = true;
01460                     }
01461
01462                     // Set cleanup registered
01463                     CleanupRegistered = true;
01464
01465                     // Set cleanup lock
01466                     lock (CleanupLock)
01467                     {
01468                         CleanupRegistered = true;
01469                     }
01470
01471                     // Set cleanup registered
01472                     CleanupRegistered = true;
01473
01474                     // Set cleanup lock
01475                     lock (CleanupLock)
01476                     {
01477                         CleanupRegistered = true;
01478                     }
01479
01480                     // Set cleanup registered
01481                     CleanupRegistered = true;
01482
01483                     // Set cleanup lock
01484                     lock (CleanupLock)
01485                     {
01486                         CleanupRegistered = true;
01487                     }
01488
01489                     // Set cleanup registered
01490                     CleanupRegistered = true;
01491
01492                     // Set cleanup lock
01493                     lock (CleanupLock)
01494                     {
01495                         CleanupRegistered = true;
01496                     }
01497
01498                     // Set cleanup registered
01499                     CleanupRegistered = true;
01500
01501                     // Set cleanup lock
01502                     lock (CleanupLock)
01503                     {
01504                         CleanupRegistered = true;
01505                     }
01506
01507                     // Set cleanup registered
01508                     CleanupRegistered = true;
01509
01510                     // Set cleanup lock
01511                     lock (CleanupLock)
01512                     {
01513                         CleanupRegistered = true;
01514                     }
01515
01516                     // Set cleanup registered
01517                     CleanupRegistered = true;
01518
01519                     // Set cleanup lock
01520                     lock (CleanupLock)
01521                     {
01522                         CleanupRegistered = true;
01523                     }
01524
01525                     // Set cleanup registered
01526                     CleanupRegistered = true;
01527
01528                     // Set cleanup lock
01529                     lock (CleanupLock)
01530                     {
01531                         CleanupRegistered = true;
01532                     }
01533
01534                     // Set cleanup registered
01535                     CleanupRegistered = true;
01536
01537                     // Set cleanup lock
01538                     lock (CleanupLock)
01539                     {
01540                         CleanupRegistered = true;
01541                     }
01542
01543                     // Set cleanup registered
01544                     CleanupRegistered = true;
01545
01546                     // Set cleanup lock
01547                     lock (CleanupLock)
01548                     {
01549                         CleanupRegistered = true;
01550                     }
01551
01552                     // Set cleanup registered
01553                     CleanupRegistered = true;
01554
01555                     // Set cleanup lock
01556                     lock (CleanupLock)
01557                     {
01558                         CleanupRegistered = true;
01559                     }
01560
01561                     // Set cleanup registered
01562                     CleanupRegistered = true;
01563
01564                     // Set cleanup lock
01565                     lock (CleanupLock)
01566                     {
01567                         CleanupRegistered = true;
01568                     }
01569
01570                     // Set cleanup registered
01571                     CleanupRegistered = true;
01572
01573                     // Set cleanup lock
01574                     lock (CleanupLock)
01575                     {
01576                         CleanupRegistered = true;
01577                     }
01578
01579                     // Set cleanup registered
01580                     CleanupRegistered = true;
01581
01582                     // Set cleanup lock
01583                     lock (CleanupLock)
01584                     {
01585                         CleanupRegistered = true;
01586                     }
01587
01588                     // Set cleanup registered
01589                     CleanupRegistered = true;
01590
01591                     // Set cleanup lock
01592                     lock (CleanupLock)
01593                     {
01594                         CleanupRegistered = true;
01595                     }
01596
01597                     // Set cleanup registered
01598                     CleanupRegistered = true;
01599
01600                     // Set cleanup lock
01601                     lock (CleanupLock)
01602                     {
01603                         CleanupRegistered = true;
01604                     }
01605
01606                     // Set cleanup registered
01607                     CleanupRegistered = true;
01608
01609                     // Set cleanup lock
01610                     lock (CleanupLock)
01611                     {
01612                         CleanupRegistered = true;
01613                     }
01614
01615                     // Set cleanup registered
01616                     CleanupRegistered = true;
01617
01618                     // Set cleanup lock
01619                     lock (CleanupLock)
01620                     {
01621                         CleanupRegistered = true;
01622                     }
01623
01624                     // Set cleanup registered
01625                     CleanupRegistered = true;
01626
01627                     // Set cleanup lock
01628                     lock (CleanupLock)
01629                     {
01630                         CleanupRegistered = true;
01631                     }
01632
01633                     // Set cleanup registered
01634                     CleanupRegistered = true;
01635
01636                     // Set cleanup lock
01637                     lock (CleanupLock)
01638                     {
01639                         CleanupRegistered = true;
01640                     }
01641
01642                     // Set cleanup registered
01643                     CleanupRegistered = true;
01644
01645                     // Set cleanup lock
01646                     lock (CleanupLock)
01647                     {
01648                         CleanupRegistered = true;
01649                     }
01650
01651                     // Set cleanup registered
01652                     CleanupRegistered = true;
01653
01654                     // Set cleanup lock
01655                     lock (CleanupLock)
01656                     {
01657                         CleanupRegistered = true;
01658                     }
01659
01660                     // Set cleanup registered
01661                     CleanupRegistered = true;
01662
01663                     // Set cleanup lock
01664                     lock (CleanupLock)
01665                     {
01666                         CleanupRegistered = true;
01667                     }
01668
01669                     // Set cleanup registered
01670                     CleanupRegistered = true;
01671
01672                     // Set cleanup lock
01673                     lock (CleanupLock)
01674                     {
01675                         CleanupRegistered = true;
01676                     }
01677
01678                     // Set cleanup registered
01679                     CleanupRegistered = true;
01680
01681                     // Set cleanup lock
01682                     lock (CleanupLock)
01683                     {
01684                         CleanupRegistered = true;
01685                     }
01686
01687                     // Set cleanup registered
01688                     CleanupRegistered = true;
01689
01690                     // Set cleanup lock
01691                     lock (CleanupLock)
01692                     {
01693                         CleanupRegistered = true;
01694                     }
01695
01696                     // Set cleanup registered
01697                     CleanupRegistered = true;
01698
01699                     // Set cleanup lock
01700                     lock (CleanupLock)
01701                     {
01702                         CleanupRegistered = true;
01703                     }
01704
01705                     // Set cleanup registered
01706                     CleanupRegistered = true;
01707
01708                     // Set cleanup lock
01709                     lock (CleanupLock)
01710                     {
01711                         CleanupRegistered = true;
01712                     }
01713
01714                     // Set cleanup registered
01715                     CleanupRegistered = true;
01716
01717                     // Set cleanup lock
01718                     lock (CleanupLock)
01719                     {
01720                         CleanupRegistered = true;
01721                     }
01722
01723                     // Set cleanup registered
01724                     CleanupRegistered = true;
01725
01726                     // Set cleanup lock
01727                     lock (CleanupLock)
01728                     {
01729                         CleanupRegistered = true;
01730                     }
01731
01732                     // Set cleanup registered
01733                     CleanupRegistered = true;
01734
01735                     // Set cleanup lock
01736                     lock (CleanupLock)
01737                     {
01738                         CleanupRegistered = true;
01739                     }
01740
01741                     // Set cleanup registered
01742                     CleanupRegistered = true;
01743
01744                     // Set cleanup lock
01745                     lock (CleanupLock)
01746                     {
01747                         CleanupRegistered = true;
01748                     }
01749
01750                     // Set cleanup registered
01751                     CleanupRegistered = true;
01752
01753                     // Set cleanup lock
01754                     lock (CleanupLock)
01755                     {
01756                         CleanupRegistered = true;
01757                     }
01758
01759                     // Set cleanup registered
01760                     CleanupRegistered = true;
01761
01762                     // Set cleanup lock
01763                     lock (CleanupLock)
01764                     {
01765                         CleanupRegistered = true;
01766                     }
01767
01768                     // Set cleanup registered
01769                     CleanupRegistered = true;
01770
01771                     // Set cleanup lock
01772                     lock (CleanupLock)
01773                     {
01774                         CleanupRegistered = true;
01775                     }
01776
01777                     // Set cleanup registered
01778                     CleanupRegistered = true;
01779
01780                     // Set cleanup lock
01781                     lock (CleanupLock)
01782                     {
01783                         CleanupRegistered = true;
01784                     }
01785
01786                     // Set cleanup registered
01787                     CleanupRegistered = true;
01788
01789                     // Set cleanup lock
01790                     lock (CleanupLock)
01791                     {
01792                         CleanupRegistered = true;
01793                     }
01794
01795                     // Set cleanup registered
01796                     CleanupRegistered = true;
01797
01798                     // Set cleanup lock
01799                     lock (CleanupLock)
01800                     {
01801                         CleanupRegistered = true;
01802                     }
01803
01804                     // Set cleanup registered
01805                     CleanupRegistered = true;
01806
01807                     // Set cleanup lock
01808                     lock (CleanupLock)
01809                     {
01810                         CleanupRegistered = true;
01811                     }
01812
01813                     // Set cleanup registered
01814                     CleanupRegistered = true;
01815
01816                     // Set cleanup lock
01817                     lock (CleanupLock)
01818                     {
01819                         CleanupRegistered = true;
01820                     }
01821
01822                     // Set cleanup registered
01823                     CleanupRegistered = true;
01824
01825                     // Set cleanup lock
01826                     lock (CleanupLock)
01827                     {
01828                         CleanupRegistered = true;
01829                     }
01830
01831                     // Set cleanup registered
01832                     CleanupRegistered = true;
01833
01834                     // Set cleanup lock
01835                     lock (CleanupLock)
01836                     {
01837                         CleanupRegistered = true;
01838                     }
01839
01840                     // Set cleanup registered
01841                     CleanupRegistered = true;
01842
01843                     // Set cleanup lock
01844                     lock (CleanupLock)
01845                     {
01846                         CleanupRegistered = true;
01847                     }
01848
01849                     // Set cleanup registered
01850                     CleanupRegistered = true;
01851
01852                     // Set cleanup lock
01853                     lock (CleanupLock)
01854                     {
01855                         CleanupRegistered = true;
01856                     }
01857
01858                     // Set cleanup registered
01859                     CleanupRegistered = true;
01860
01861                     // Set cleanup lock
01862                     lock (CleanupLock)
01863                     {
01864                         CleanupRegistered = true;
01865                     }
01866
01867                     // Set cleanup registered
01868                     CleanupRegistered = true;
01869
01870                     // Set cleanup lock
01871                     lock (CleanupLock)
01872                     {
01873                         CleanupRegistered = true;
01874                     }
01875
01876                     // Set cleanup registered
01877                     CleanupRegistered = true;
01878
01879                     // Set cleanup lock
01880                     lock (CleanupLock)
01881                     {
01882                         CleanupRegistered = true;
01883                     }
01884
01885                     // Set cleanup registered
01886                     CleanupRegistered = true;
01887
01888                     // Set cleanup lock
01889                     lock (CleanupLock)
01890                     {
01891                         CleanupRegistered = true;
01892                     }
01893
01894                     // Set cleanup registered
01895                     CleanupRegistered = true;
01896
01897                     // Set cleanup lock
01898                     lock (CleanupLock)
01899                     {
01900                         CleanupRegistered = true;
01901                     }
01902
01903                     // Set cleanup registered
01904                     CleanupRegistered = true;
01905
01906                     // Set cleanup lock
01907                     lock (CleanupLock)
01908                     {
01909                         CleanupRegistered = true;
01910                     }
01911
01912                     // Set cleanup registered
01913                     CleanupRegistered = true;
01914
01915                     // Set cleanup lock
01916                     lock (CleanupLock)
01917                     {
01918                         CleanupRegistered = true;
01919                     }
01920
01921                     // Set cleanup registered
01922                     CleanupRegistered = true;
01923
01924                     // Set cleanup lock
01925                     lock (CleanupLock)
01926                     {
01927                         CleanupRegistered = true;
01928                     }
01
```

```

00810                     catch { }
00811                 }
00812             }
00813             using (StreamReader reader = new StreamReader(client))
00814             {
00815                 while (true)
00816                 {
00817                     string message = reader.ReadLine();
00818
00819                     if (!string.IsNullOrEmpty(message))
00820                     {
00821                         StandardOutputMessage?.Invoke(null, new
00822                         MessageEventArgs(message));
00823                     }
00824                 }
00825             }
00826         });
00827
00828         // Start stderr pipe (this is actually a socket)
00829         _ = Task.Run(() =>
00830         {
00831             using (NamedPipeClientStream client = new NamedPipeClientStream(pipeName +
00832             "-err"))
00833             {
00834                 while (true)
00835                 {
00836                     try
00837                     {
00838                         client.Connect(100);
00839                         break;
00840                     }
00841                     catch { }
00842                 }
00843
00844                 using (StreamReader reader = new StreamReader(client))
00845                 {
00846                     while (true)
00847                     {
00848                         string message = reader.ReadLine();
00849
00850                         if (!string.IsNullOrEmpty(message))
00851                         {
00852                             StandardErrorMessage?.Invoke(null, new MessageEventArgs(message));
00853                         }
00854                     }
00855                 }
00856             }
00857         });
00858     });
00859
00860     await redirectOutputTask;
00861
00862     ConsoleOut = Console.Out;
00863     ConsoleErr = Console.Error;
00864
00865     ConsoleColor fg = Console.ForegroundColor;
00866     ConsoleColor bg = Console.BackgroundColor;
00867
00868     Console.ResetColor();
00869
00870     DefaultForeground = Console.ForegroundColor;
00871     DefaultBackground = Console.BackgroundColor;
00872
00873     Console.ForegroundColor = fg;
00874     Console.BackgroundColor = bg;
00875
00876     Console.SetOut(new FileDescriptorTextWriter(Console.Out.Encoding, StdOutFD));
00877     Console.SetError(new FileDescriptorTextWriter(Console.Error.Encoding, StdErrFD));
00878     }
00879 }
00880
00881 private static async Task RedirectOutputWindows()
00882 {
00883     if (StdOutFD < 0 && StdErrFD < 0)
00884     {
00885         string pipeName = "MuPDFCore-" + Guid.NewGuid().ToString();
00886
00887         Task redirectOutputTask = System.Threading.Tasks.Task.Run(() =>
00888         {
00889             NativeMethods.RedirectOutput(out StdOutFD, out StdErrFD, "\\\\".\\pipe\\" +
00890             pipeName + "-out", "\\\\".\\pipe\\" + pipeName + "-err");
00891         });
00892
00893         // Start stdout pipe
00894         _ = Task.Run(() =>

```

```
00894             {
00895                 using (NamedPipeClientStream client = new NamedPipeClientStream(pipeName +
00896                     "-out"))
00897                 {
00898                     while (true)
00899                     {
00900                         try
00901                         {
00902                             client.Connect(100);
00903                             break;
00904                         }
00905                         catch { }
00906                     }
00907                     using (StreamReader reader = new StreamReader(client))
00908                     {
00909                         while (true)
00910                         {
00911                             string message = reader.ReadLine();
00912                             if (!string.IsNullOrEmpty(message))
00913                             {
00914                                 StandardOutputMessage?.Invoke(null, new
00915                         MessageEventArgs(message));
00916                             }
00917                         }
00918                     }
00919                 }
00920             );
00921
00922             // Start stderr pipe
00923             _ = Task.Run(() =>
00924             {
00925                 using (NamedPipeClientStream client = new NamedPipeClientStream(pipeName +
00926                     "-err"))
00927                 {
00928                     while (true)
00929                     {
00930                         try
00931                         {
00932                             client.Connect(100);
00933                             break;
00934                         }
00935                         catch { }
00936                     }
00937
00938                     using (StreamReader reader = new StreamReader(client))
00939                     {
00940                         while (true)
00941                         {
00942                             string message = reader.ReadLine();
00943                             if (!string.IsNullOrEmpty(message))
00944                             {
00945                                 StandardErrorMessage?.Invoke(null, new MessageEventArgs(message));
00946                             }
00947                         }
00948                     }
00949                 }
00950             });
00951
00952         });
00953
00954         await redirectOutputTask;
00955
00956         ConsoleOut = Console.Out;
00957         ConsoleErr = Console.Error;
00958
00959         ConsoleColor fg = Console.ForegroundColor;
00960         ConsoleColor bg = Console.BackgroundColor;
00961
00962         Console.ResetColor();
00963
00964         DefaultForeground = Console.ForegroundColor;
00965         DefaultBackground = Console.BackgroundColor;
00966
00967         Console.ForegroundColor = fg;
00968         Console.BackgroundColor = bg;
00969
00970         Console.SetOut(new FileDescriptorTextWriter(Console.Out.Encoding, StdOutFD));
00971         Console.SetError(new FileDescriptorTextWriter(Console.Error.Encoding, StdErrFD));
00972     }
00973 }
00974
00975 /// <summary>
00976 /// Reset the default standard output and error streams for the native MuPDF library.
00977 /// </summary>
```

```
00978     public static void ResetOutput()
00979     {
00980         lock (CleanupLock)
00981         {
00982             if (StdOutFD >= 0 && StdErrFD >= 0)
00983             {
00984                 NativeMethods.ResetOutput(StdOutFD, StdErrFD);
00985
00986                 Console.SetOut(ConsoleOut);
00987                 Console.SetError(ConsoleErr);
00988
00989                 StdOutFD = -1;
00990                 StdErrFD = -1;
00991
00992             if
00993             (!System.Runtime.InteropServices.RuntimeInformation.IsOSPlatform(OSPlatform.Windows))
00994             {
00995                 File.Delete(PipeName + "-out");
00996                 File.Delete(PipeName + "-err");
00997             }
00998         }
00999     }
01000
01001     internal class FileDescriptorTextWriter : TextWriter
01002     {
01003         public override Encoding Encoding { get; }
01004         private int FileDescriptor { get; }
01005
01006         public FileDescriptorTextWriter(Encoding encoding, int fileDescriptor)
01007         {
01008             this.Encoding = encoding;
01009             this.FileDescriptor = fileDescriptor;
01010         }
01011
01012         public override void Write(string value)
01013         {
01014             StringBuilder sb = new StringBuilder();
01015
01016             if (Console.ForegroundColor != DefaultForeground || Console.BackgroundColor != DefaultBackground)
01017             {
01018                 sb.Append(" [");
01019             }
01020
01021             if (Console.ForegroundColor != DefaultForeground)
01022             {
01023                 switch (Console.ForegroundColor)
01024                 {
01025                     case ConsoleColor.Black:
01026                         sb.Append("30");
01027                         break;
01028                     case ConsoleColor.DarkRed:
01029                         sb.Append("31");
01030                         break;
01031                     case ConsoleColor.DarkGreen:
01032                         sb.Append("32");
01033                         break;
01034                     case ConsoleColor.DarkYellow:
01035                         sb.Append("33");
01036                         break;
01037                     case ConsoleColor.DarkBlue:
01038                         sb.Append("34");
01039                         break;
01040                     case ConsoleColor.DarkMagenta:
01041                         sb.Append("35");
01042                         break;
01043                     case ConsoleColor.DarkCyan:
01044                         sb.Append("36");
01045                         break;
01046                     case ConsoleColor.Gray:
01047                         sb.Append("37");
01048                         break;
01049                     case ConsoleColor.DarkGray:
01050                         sb.Append("90");
01051                         break;
01052                     case ConsoleColor.Red:
01053                         sb.Append("91");
01054                         break;
01055                     case ConsoleColor.Green:
01056                         sb.Append("92");
01057                         break;
01058                     case ConsoleColor.Yellow:
01059                         sb.Append("93");
01060                         break;
01061                     case ConsoleColor.Blue:
01062                         sb.Append("94");
```

```
01063             break;
01064         case ConsoleColor.Magenta:
01065             sb.Append("95");
01066             break;
01067         case ConsoleColor.Cyan:
01068             sb.Append("96");
01069             break;
01070         case ConsoleColor.White:
01071             sb.Append("97");
01072             break;
01073     }
01074 }
01075
01076 if (Console.ForegroundColor != DefaultForeground && Console.BackgroundColor !=
DefaultBackground)
01077 {
01078     sb.Append(";");
01079 }
01080
01081 if (Console.BackgroundColor != DefaultBackground)
01082 {
01083     switch (Console.BackgroundColor)
01084 {
01085         case ConsoleColor.Black:
01086             sb.Append("40");
01087             break;
01088         case ConsoleColor.DarkRed:
01089             sb.Append("41");
01090             break;
01091         case ConsoleColor.DarkGreen:
01092             sb.Append("42");
01093             break;
01094         case ConsoleColor.DarkYellow:
01095             sb.Append("43");
01096             break;
01097         case ConsoleColor.DarkBlue:
01098             sb.Append("44");
01099             break;
01100         case ConsoleColor.DarkMagenta:
01101             sb.Append("45");
01102             break;
01103         case ConsoleColor.DarkCyan:
01104             sb.Append("46");
01105             break;
01106         case ConsoleColor.Gray:
01107             sb.Append("47");
01108             break;
01109         case ConsoleColor.DarkGray:
01110             sb.Append("100");
01111             break;
01112         case ConsoleColor.Red:
01113             sb.Append("101");
01114             break;
01115         case ConsoleColor.Green:
01116             sb.Append("102");
01117             break;
01118         case ConsoleColor.Yellow:
01119             sb.Append("103");
01120             break;
01121         case ConsoleColor.Blue:
01122             sb.Append("104");
01123             break;
01124         case ConsoleColor.Magenta:
01125             sb.Append("105");
01126             break;
01127         case ConsoleColor.Cyan:
01128             sb.Append("106");
01129             break;
01130         case ConsoleColor.White:
01131             sb.Append("107");
01132             break;
01133     }
01134 }
01135
01136 if (Console.ForegroundColor != DefaultForeground || Console.BackgroundColor !=
DefaultBackground)
01137 {
01138     sb.Append("m");
01139 }
01140
01141     sb.Append(value);
01142
01143 if (Console.ForegroundColor != DefaultForeground || Console.BackgroundColor !=
DefaultBackground)
01144 {
01145     sb.Append(" [");
01146 }
```

```

01147             if (Console.ForegroundColor != DefaultForeground)
01148             {
01149                 sb.Append("39");
01150             }
01151         }
01152     }
01153     if (Console.ForegroundColor != DefaultForeground && Console.BackgroundColor != DefaultBackground)
01154     {
01155         sb.Append(";");
01156     }
01157     if (Console.BackgroundColor != DefaultBackground)
01158     {
01159         sb.Append("49");
01160     }
01161 }
01162     if (Console.ForegroundColor != DefaultForeground || Console.BackgroundColor != DefaultBackground)
01163     {
01164         sb.Append("m");
01165     }
01166 }
01167 NativeMethods.WriteToFileDescriptor(FileDescriptor, sb.ToString(), sb.Length);
01168 }
01169
01170 public override void Write(char value)
01171 {
01172     Write(value.ToString());
01173 }
01174
01175 public override void Write(char[] buffer)
01176 {
01177     Write(new string(buffer));
01178 }
01179
01180
01181 public override void Write(char[] buffer, int index, int count)
01182 {
01183     Write(new string(buffer, index, count));
01184 }
01185
01186 public override void WriteLine(string value)
01187 {
01188     Write(value);
01189     WriteLine();
01190 }
01191 }
01192 }
01193
01194 /// <summary>
01195 /// Native methods.
01196 /// </summary>
01197 internal static class NativeMethods
01198 {
01199 /// <summary>
01200 /// Create a MuPDF context object with the specified store size.
01201 /// </summary>
01202 /// <param name="store_size">Maximum size in bytes of the resource store.</param>
01203 /// <param name="out_ctx">A pointer to the native context object.</param>
01204 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors occurred.</returns>
01205     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01206     internal static extern int CreateContext(ulong store_size, ref IntPtr out_ctx);
01207
01208 /// <summary>
01209 /// Free a context and its global store.
01210 /// </summary>
01211 /// <param name="ctx">A pointer to the native context to free.</param>
01212 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors occurred.</returns>
01213     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01214     internal static extern int DisposeContext(IntPtr ctx);
01215
01216 /// <summary>
01217 /// Evict items from the store until the total size of the objects in the store is reduced to a given percentage of its current size.
01218 /// </summary>
01219 /// <param name="ctx">The context whose store should be shrunk.</param>
01220 /// <param name="perc">Fraction of current size to reduce the store to.</param>
01221 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors occurred.</returns>
01222     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01223     internal static extern int ShrinkStore(IntPtr ctx, uint perc);
01224
01225 /// <summary>
01226 /// Evict every item from the store.
01227 /// </summary>

```

```

01228 /// <param name="ctx">The context whose store should be emptied.</param>
01229 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01230     occurred.</returns>
01231     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01232     internal static extern void EmptyStore(IntPtr ctx);
01233 
01234 /// <summary>
01235 /// Get the current size of the store.
01236 /// </summary>
01237 /// <param name="ctx">The context whose store's size should be determined.</param>
01238 /// <returns>The current size in bytes of the store.</returns>
01239     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01240     internal static extern ulong GetCurrentStoreSize(IntPtr ctx);
01241 
01242 /// <summary>
01243 /// Get the maximum size of the store.
01244 /// </summary>
01245 /// <param name="ctx">The context whose store's maximum size should be determined.</param>
01246 /// <returns>The maximum size in bytes of the store.</returns>
01247     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01248     internal static extern ulong GetMaxStoreSize(IntPtr ctx);
01249 
01250 /// <summary>
01251 /// Set the current antialiasing levels.
01252 /// <param name="ctx">The context whose antialiasing levels should be set.</param>
01253 /// <param name="aa">The overall antialiasing level. Ignored if <code>0</code>.</param>
01254 /// <param name="graphics_aa">The graphics antialiasing level. Ignored if <code>0</code>.</param>
01255 /// <param name="text_aa">The text antialiasing level. Ignored if <code>0</code>.</param>
01256     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01257     internal static extern void SetAALevel(IntPtr ctx, int aa, int graphics_aa, int text_aa);
01258 
01259 /// <summary>
01260 /// Get the current antialiasing levels.
01261 /// </summary>
01262 /// <param name="ctx">The context whose antialiasing levels should be retrieved.</param>
01263 /// <param name="out_aa">The overall antialiasing level.</param>
01264 /// <param name="out_graphics_aa">The graphics antialiasing level.</param>
01265 /// <param name="out_text_aa">The text antialiasing level.</param>
01266     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01267     internal static extern void GetAALevel(IntPtr ctx, out int out_aa, out int out_graphics_aa,
01268     out int out_text_aa);
01269 
01270 /// <summary>
01271 /// Create a display list from a page.
01272 /// <param name="ctx">A pointer to the context used to create the document.</param>
01273 /// <param name="page">A pointer to the page that should be used to create the display list.</param>
01274 /// <param name="annotations">An integer indicating whether annotations should be included in the
01275     display list (1) or not (any other value).</param>
01276 /// <param name="out_display_list">A pointer to the newly-created display list.</param>
01277 /// <param name="out_x0">The left coordinate of the display list's bounds.</param>
01278 /// <param name="out_y0">The top coordinate of the display list's bounds.</param>
01279 /// <param name="out_x1">The right coordinate of the display list's bounds.</param>
01280 /// <param name="out_y1">The bottom coordinate of the display list's bounds.</param>
01281 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01282     occurred.</returns>
01283     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01284     internal static extern int GetDisplayList(IntPtr ctx, IntPtr page, int annotations, ref IntPtr
01285     out_display_list, ref float out_x0, ref float out_y0, ref float out_x1, ref float out_y1);
01286 
01287 /// <summary>
01288 /// Free a display list.
01289 /// </summary>
01290 /// <param name="ctx">The context that was used to create the display list.</param>
01291 /// <param name="list">The display list to dispose.</param>
01292 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01293     occurred.</returns>
01294     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01295     internal static extern int DisposeDisplayList(IntPtr ctx, IntPtr list);
01296 
01297 /// <summary>
01298 /// Create a new document from a stream.
01299 /// </summary>
01300 /// <param name="ctx">The context to which the document will belong.</param>
01301 /// <param name="data">A pointer to a byte array containing the data that makes up the
01302     document.</param>
01303 /// <param name="data_length">The length in bytes of the data that makes up the document.</param>
01304 /// <param name="file_type">The type (extension) of the document.</param>
01305 /// <param name="get_image_resolution">If this is not 0, try opening the stream as an image and return
01306     the actual resolution (in DPI) of the image. Otherwise (or if trying to open the stream as an image
01307     fails), the returned resolution will be -1.</param>
01308 /// <param name="out_doc">The newly created document.</param>
01309 /// <param name="out_str">The newly created stream (so that it can be disposed later).</param>
01310 /// <param name="out_page_count">The number of pages in the document.</param>
01311 /// <param name="out_image_xres">If the document is an image file, the horizontal resolution of the
01312     image.</param>

```

```

01305 /// <param name="out_image_yres">If the document is an image file, the vertical resolution of the
01306     image.</param>
01307     /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01308     occurred.</returns>
01309     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01310     internal static extern int CreateDocumentFromStream(IntPtr ctx, IntPtr data, ulong
01311         data_length, string file_type, int get_image_resolution, ref IntPtr out_doc, ref IntPtr out_str, ref
01312         int out_page_count, ref float out_image_xres, ref float out_image_yres);
01313     /// <summary>
01314     /// Create a new document from a file name.
01315     /// </summary>
01316     /// <param name="ctx">The context to which the document will belong.</param>
01317     /// <param name="file_name">The path of the file to open, UTF-8 encoded.</param>
01318     /// <param name="get_image_resolution">If this is not 0, try opening the file as an image and return
01319     the actual resolution (in DPI) of the image. Otherwise (or if trying to open the file as an image
01320     fails), the returned resolution will be -1.</param>
01321     /// <param name="out_doc">The newly created document.</param>
01322     /// <param name="out_page_count">The number of pages in the document.</param>
01323     /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01324     occurred.</returns>
01325     /// <param name="out_image_xres">If the document is an image file, the horizontal resolution of the
01326     image.</param>
01327     /// <param name="out_image_yres">If the document is an image file, the vertical resolution of the
01328     image.</param>
01329     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01330     internal static extern int CreateDocumentFromFile(IntPtr ctx, IntPtr file_name, int
01331         get_image_resolution, ref IntPtr out_doc, ref int out_page_count, ref float out_image_xres, ref float
01332         out_image_yres);
01333     /// <summary>
01334     /// Free a stream and its associated resources.
01335     /// </summary>
01336     /// <param name="ctx">The context that was used while creating the stream.</param>
01337     /// <param name="str">The stream to free.</param>
01338     /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01339     occurred.</returns>
01340     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01341     internal static extern int DisposeStream(IntPtr ctx, IntPtr str);
01342     /// <summary>
01343     /// Free a document and its associated resources.
01344     /// </summary>
01345     /// <param name="ctx">The context that was used in creating the document.</param>
01346     /// <param name="doc">The document to free.</param>
01347     /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01348     occurred.</returns>
01349     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01350     internal static extern int DisposeDocument(IntPtr ctx, IntPtr doc);
01351     /// <summary>
01352     /// Render (part of) a display list to an array of bytes starting at the specified pointer.
01353     /// </summary>
01354     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01355     /// <param name="list">The display list to render.</param>
01356     /// <param name="x0">The left coordinate in page units of the region of the display list that should
01357     be rendererd.</param>
01358     /// <param name="y0">The top coordinate in page units of the region of the display list that should be
01359     rendererd.</param>
01360     /// <param name="x1">The right coordinate in page units of the region of the display list that should
01361     be rendererd.</param>
01362     /// <param name="y1">The bottom coordinate in page units of the region of the display list that should
01363     be rendererd.</param>
01364     /// <param name="zoom">How much the specified region should be scaled when rendering. This determines
01365     the size in pixels of the rendered image.</param>
01366     /// <param name="colorFormat">The pixel data format.</param>
01367     /// <param name="pixel_storage">A pointer indicating where the pixel bytes will be written. There
01368     must be enough space available!</param>
01369     /// <param name="cookie">A pointer to a cookie object that can be used to track progress and/or abort
01370     rendering. Can be null.</param>
01371     /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01372     occurred.</returns>
01373     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01374     internal static extern int RenderSubDisplayList(IntPtr ctx, IntPtr list, float x0, float y0,
01375         float x1, float y1, float zoom, int colorFormat, IntPtr pixel_storage, IntPtr cookie);
01376     /// <summary>
01377     /// Get the specified bounding box from a page.
01378     /// </summary>
01379     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01380     /// <param name="page">The page whose bounding box should be extracted.</param>
01381     /// <param name="box">An integer equivalent to <see cref="BoxType"/> specifying the box to
01382     extract.</param>
01383     /// <param name="out_x">When this method returns, this variable will contain the left coordinate of
01384     the bounding box.</param>
01385     /// <param name="out_y">When this method returns, this variable will contain the top coordinate of the
01386     bounding box.</param>

```

```

01367 /// <param name="out_w">When this method returns, this variable will contain the width of the bounding
01368 //</param>
01369 /// <param name="out_h">When this method returns, this variable will contain the height of the
01370 //</param>
01371 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01372 // occurred.</returns>
01373 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01374     internal static extern int GetPageBox(IntPtr ctx, IntPtr page, int box, ref float out_x, ref
01375 float out_y, ref float out_w, ref float out_h);
01376 /// <summary>
01377 /// Load a page from a document.
01378 /// </summary>
01379 /// <param name="ctx">The context to which the document belongs.</param>
01380 /// <param name="doc">The document from which the page should be extracted.</param>
01381 /// <param name="page_number">The page number.</param>
01382 /// <param name="out_page">The newly extracted page.</param>
01383 /// <param name="out_x">The left coordinate of the page's bounds.</param>
01384 /// <param name="out_y">The top coordinate of the page's bounds.</param>
01385 /// <param name="out_w">The width of the page.</param>
01386 /// <param name="out_h">The height of the page.</param>
01387 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01388 // occurred.</returns>
01389 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01390     internal static extern int LoadPage(IntPtr ctx, IntPtr doc, int page_number, ref IntPtr
01391 out_page, ref float out_x, ref float out_y, ref float out_w, ref float out_h);
01392 /// <summary>
01393 /// Free a page and its associated resources.
01394 /// </summary>
01395 /// <param name="ctx">The context to which the document containing the page belongs.</param>
01396 /// <param name="page">The page to free.</param>
01397 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01398 // occurred.</returns>
01399 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01400     internal static extern int DisposePage(IntPtr ctx, IntPtr page);
01401 /// <summary>
01402 /// Layout ref流able document types.
01403 /// </summary>
01404 /// <param name="ctx">The context to which the document belongs.</param>
01405 /// <param name="doc">The document to layout.</param>
01406 /// <param name="width">The page width.</param>
01407 /// <param name="height">The page height.</param>
01408 /// <param name="em">The default font size, in points.</param>
01409 /// <param name="out_page_count">The number of pages in the document, after the layout.</param>
01410 /// <returns>An integer detailing whether any errors occurred.</returns>
01411 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01412     internal static extern int LayoutDocument(IntPtr ctx, IntPtr doc, float width, float height,
01413 float em, out int out_page_count);
01414 /// <summary>
01415 /// Create cloned contexts that can be used in multithreaded rendering.
01416 /// </summary>
01417 /// <param name="ctx">The original context to clone.</param>
01418 /// <param name="count">The number of cloned contexts to create.</param>
01419 /// <param name="out_contexts">An array of pointers to the cloned contexts.</param>
01420 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01421 // occurred.</returns>
01422 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01423     internal static extern int CloneContext(IntPtr ctx, int count, IntPtr out_contexts);
01424 /// <summary>
01425 /// Save (part of) a display list to an image file in the specified format.
01426 /// </summary>
01427 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01428 /// <param name="list">The display list to render.</param>
01429 /// <param name="x0">The left coordinate in page units of the region of the display list that should
01430 // be rendererd.</param>
01431 /// <param name="y0">The top coordinate in page units of the region of the display list that should be
01432 // rendererd.</param>
01433 /// <param name="x1">The right coordinate in page units of the region of the display list that should
01434 // be rendererd.</param>
01435 /// <param name="y1">The bottom coordinate in page units of the region of the display list that should
01436 // be rendererd.</param>
01437 /// <param name="zoom">How much the specified region should be scaled when rendering. This determines
01438 // the size in pixels of the rendered image.</param>
01439 /// <param name="colorFormat">The pixel data format.</param>
01440 /// <param name="file_name">The path to the output file, UTF-8 encoded.</param>
01441 /// <param name="output_format">An integer equivalent to <see cref="RasterOutputFileTypes"/>
01442 // specifying the output format.</param>
01443 /// <param name="quality">Quality level for the output format (where applicable).</param>
01444 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01445 // occurred.</returns>
01446 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01447     internal static extern int SaveImage(IntPtr ctx, IntPtr list, float x0, float y0, float x1,
01448 float y1, float zoom, int colorFormat, IntPtr file_name, int output_format, int quality);

```

```

01437 /// <summary>
01438 /// Write (part of) a display list to an image buffer in the specified format.
01439 /// </summary>
01440 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01441 /// <param name="list">The display list to render.</param>
01442 /// <param name="x0">The left coordinate in page units of the region of the display list that should
01443 /// be rendererd.</param>
01444 /// <param name="y0">The top coordinate in page units of the region of the display list that should be
01445 /// rendererd.</param>
01446 /// <param name="x1">The right coordinate in page units of the region of the display list that should
01447 /// be rendererd.</param>
01448 /// <param name="y1">The bottom coordinate in page units of the region of the display list that should
01449 /// be rendererd.</param>
01450 /// <param name="zoom">How much the specified region should be scaled when rendering. This determines
01451 /// the size in pixels of the rendered image.</param>
01452 /// <param name="colorFormat">The pixel data format.</param>
01453 /// <param name="output_format">An integer equivalent to <see cref="RasterOutputFileTypes"/>
01454 /// specifying the output format.</param>
01455 /// <param name="quality">Quality level for the output format (where applicable).</param>
01456 /// <param name="out_buffer">The address of the buffer on which the data has been written (only useful
01457 /// for disposing the buffer later).</param>
01458 /// <param name="out_data">The address of the byte array where the data has been actually
01459 /// written.</param>
01460 /// <param name="out_length">The length in bytes of the image data.</param>
01461 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01462 /// occurred.</returns>
01463 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01464     internal static extern int WriteImage(IntPtr ctx, IntPtr list, float x0, float y0, float x1,
01465         float y1, float zoom, int colorFormat, int output_format, int quality, ref IntPtr out_buffer, ref
01466         IntPtr out_data, ref ulong out_length);
01467 /// <summary>
01468 /// Free a native buffer and its associated resources.
01469 /// </summary>
01470 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01471 /// <param name="buf">The buffer to free.</param>
01472 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01473 /// occurred.</returns>
01474 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01475     internal static extern int DisposeBuffer(IntPtr ctx, IntPtr buf);
01476 /// <summary>
01477 /// Create a new document writer object.
01478 /// </summary>
01479 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01480 /// <param name="file_name">The name of file that will hold the writer's output, UTF-8
01481 /// encoded.</param>
01482 /// <param name="format">An integer equivalent to <see cref="DocumentOutputFileTypes"/> specifying the
01483 /// output format.</param>
01484 /// <param name="out_document_writer">A pointer to the new document writer object.</param>
01485 /// <param name="options">Options for the document writer.</param>
01486 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01487 /// occurred.</returns>
01488 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01489     internal static extern int CreateDocumentWriter(IntPtr ctx, IntPtr file_name, int format,
01490         IntPtr options, ref IntPtr out_document_writer);
01491 /// <summary>
01492 /// Render (part of) a display list as a page in the specified document writer.
01493 /// </summary>
01494 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01495 /// <param name="list">The display list to render.</param>
01496 /// <param name="x0">The left coordinate in page units of the region of the display list that should
01497 /// be rendererd.</param>
01498 /// <param name="y0">The top coordinate in page units of the region of the display list that should be
01499 /// rendererd.</param>
01500 /// <param name="x1">The right coordinate in page units of the region of the display list that should
01501 /// be rendererd.</param>
01502 /// <param name="y1">The bottom coordinate in page units of the region of the display list that should
01503 /// be rendererd.</param>
01504 /// <param name="zoom">How much the specified region should be scaled when rendering. This will
01505 /// determine the final size of the page.</param>
01506 /// <param name="writ">The document writer on which the page should be written.</param>
01507 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01508 /// occurred.</returns>
01509 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01510     internal static extern int WriteSubDisplayListAsPage(IntPtr ctx, IntPtr list, float x0, float
01511         y0, float x1, float y1, float zoom, IntPtr writ);
01512 /// <summary>
01513 /// Finalise a document writer, closing the file and freeing all resources.
01514 /// </summary>
01515 /// <param name="ctx">The context that was used to create the document writer.</param>
01516 /// <param name="writ">The document writer to finalise.</param>
01517 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01518 /// occurred.</returns>

```

```

01500     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01501     internal static extern int FinalizeDocumentWriter(IntPtr ctx, IntPtr writ);
01502
01503 /// <summary>
01504 /// Get the contents of a structured text character.
01505 /// </summary>
01506 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01507 /// <param name="character">The address of the character.</param>
01508 /// <param name="out_c">Unicode code point of the character.</param>
01509 /// <param name="out_color">An sRGB hex representation of the colour of the character.</param>
01510 /// <param name="out_origin_x">The x coordinate of the baseline origin of the character.</param>
01511 /// <param name="out_origin_y">The y coordinate of the baseline origin of the character.</param>
01512 /// <param name="out_size">The size in points of the character.</param>
01513 /// <param name="out_ll_x">The x coordinate of the lower left corner of the bounding quad.</param>
01514 /// <param name="out_ll_y">The y coordinate of the lower left corner of the bounding quad.</param>
01515 /// <param name="out_ul_x">The x coordinate of the upper left corner of the bounding quad.</param>
01516 /// <param name="out_ul_y">The y coordinate of the upper left corner of the bounding quad.</param>
01517 /// <param name="out_ur_x">The x coordinate of the upper right corner of the bounding quad.</param>
01518 /// <param name="out_ur_y">The y coordinate of the upper right corner of the bounding quad.</param>
01519 /// <param name="out_lr_x">The x coordinate of the lower right corner of the bounding quad.</param>
01520 /// <param name="out_lr_y">The y coordinate of the lower right corner of the bounding quad.</param>
01521 /// <param name="out_bidi">Even for LTR, odd for RTL.</param>
01522 /// <param name="out_font">Font used to draw the character.</param>
01523 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01524     occurred.</returns>
01525     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01526     internal static extern int GetStructuredTextChar(IntPtr ctx, IntPtr character, ref int out_c,
01527     ref uint out_color, ref float out_origin_x, ref float out_origin_y, ref float out_size, ref float
01528     out_ll_x, ref float out_ll_y, ref float out_ul_x, ref float out_ul_y, ref float out_ur_x, ref float
01529     out_ur_y, ref float out_lr_x, ref float out_lr_y, ref int out_bidi, ref IntPtr out_font);
01530
01531 /// <summary>
01532 /// Get an array of structured text characters from a structured text line.
01533 /// </summary>
01534 /// <param name="line">The structured text line from which the characters should be extracted.</param>
01535 /// <param name="out_chars">An array of pointers to the structured text characters.</param>
01536 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01537     occurred.</returns>
01538     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01539     internal static extern int GetStructuredTextChars(IntPtr line, IntPtr out_chars);
01540
01541 /// <summary>
01542 /// Get the contents of a structured text line.
01543 /// </summary>
01544 /// <param name="line">The address of the line.</param>
01545 /// <param name="out_wmode">An integer equivalent to <see cref="MuPDFStructuredTextLine"/>
01546     representing the writing mode of the line.</param>
01547 /// <param name="out_x0">The left coordinate in page units of the bounding box of the line.</param>
01548 /// <param name="out_y0">The top coordinate in page units of the bounding box of the line.</param>
01549 /// <param name="out_x1">The right coordinate in page units of the bounding box of the line.</param>
01550 /// <param name="out_y1">The bottom coordinate in page units of the bounding box of the line.</param>
01551 /// <param name="out_x">The x component of the normalised direction of the baseline.</param>
01552 /// <param name="out_y">The y component of the normalised direction of the baseline.</param>
01553 /// <param name="out_char_count">The number of characters in the line.</param>
01554 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01555     occurred.</returns>
01556     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01557     internal static extern int GetStructuredTextLine(IntPtr line, ref int out_wmode, ref float
01558     out_x0, ref float out_y0, ref float out_x1, ref float out_y1, ref float out_x, ref float out_y, ref
01559     int out_char_count);
01560
01561 /// <summary>
01562 /// Get an array of structured text lines from a structured text block.
01563 /// </summary>
01564 /// <param name="block">The structured text block from which the lines should be extracted.</param>
01565 /// <param name="out_lines">An array of pointers to the structured text lines.</param>
01566 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01567     occurred.</returns>
01568     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01569     internal static extern void GetStructStructuredTextBlockRawStructure(IntPtr struct_block, int
01570     raw_length, IntPtr out_raw);
01571
01572 /// <summary>
01573 /// Get information about a structural element block.
01574 /// </summary>
01575 /// <param name="struct_block">A pointer to the structural element block.</param>

```

```

01574 /// <param name="out_raw_length">When this method returns, this variable will contain the length of
01575 // the block's raw structure type.</param>
01576 /// <param name="out_standard">When this method returns, this variable will contain the block's
01577 // standard structure type.</param>
01578 /// <param name="out_parent">When this method returns, this variable will contain a pointer to the
01579 // block's structural parent (but this seems to always return <see cref="IntPtr.Zero"/>).</param>
01580 /// <param name="out_blocks">A pointer to an array of pointers that will be filled with the addresses
01581 // of the childrens of this block.</param>
01582 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01583 // occurred.</returns>
01584 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01585 internal static extern int GetStructStructuredTextBlock(IntPtr struct_block, ref int
01586 out_raw_length, ref int out_standard, ref IntPtr out_parent, IntPtr out_blocks);
01587 /// <summary>
01588 // Count the number of children within a structural element block.
01589 /// </summary>
01590 /// <param name="struct_block">A pointer to the structural element block.</param>
01591 /// <returns>The number of children within a structural element block.</returns>
01592 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01593 internal static extern int CountStructStructuredTextBlockChildren(IntPtr struct_block);
01594 /// <summary>
01595 // Get information about a grid block.
01596 /// </summary>
01597 /// <param name="block">A pointer to the grid block.</param>
01598 /// <param name="xs_len">The number of X grid lines, as returned by <see
01599 // cref="GetStructuredTextBlock"/>.</param>
01600 /// <param name="ys_len">The number of Y grid lines, as returned by <see
01601 // cref="GetStructuredTextBlock"/>.</param>
01602 /// <param name="out_x_max_uncertainty">When this method returns, this variable will contain the
01603 // maximum X uncertainty.</param>
01604 /// <param name="out_y_max_uncertainty">When this method returns, this variable will contain the
01605 // maximum Y uncertainty.</param>
01606 /// <param name="out_x_pos">A pointer to an array of <see cref="float"/>s, which will be filled by
01607 // this method.</param>
01608 /// <param name="out_y_pos">A pointer to an array of <see cref="float"/>s, which will be filled by
01609 // this method.</param>
01610 /// <param name="out_x_uncertainty">A pointer to an array of <see cref="int"/>s, which will be filled
01611 // by this method.</param>
01612 /// <param name="out_y_uncertainty">A pointer to an array of <see cref="int"/>s, which will be filled
01613 // by this method.</param>
01614 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01615 // occurred.</returns>
01616 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01617 internal static extern int GetGridStructuredTextBlock(IntPtr block, int xs_len, int ys_len,
01618 ref int out_x_max_uncertainty, ref int out_y_max_uncertainty, IntPtr out_x_pos, IntPtr out_y_pos,
01619 IntPtr out_x_uncertainty, IntPtr out_y_uncertainty);
01620 /// <summary>
01621 // Get the contents of a structured text block.
01622 /// </summary>
01623 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01624 /// <param name="block">The address of the block.</param>
01625 /// <param name="out_type">An integer equivalent to <see cref="MuPDFStructuredTextBlock.Types"/>
01626 // representing the type of the block.</param>
01627 /// <param name="out_x0">The left coordinate in page units of the bounding box of the block.</param>
01628 /// <param name="out_y0">The top coordinate in page units of the bounding box of the block.</param>
01629 /// <param name="out_x1">The right coordinate in page units of the bounding box of the block.</param>
01630 /// <param name="out_y1">The bottom coordinate in page units of the bounding box of the block.</param>
01631 /// <param name="out_line_count">The number of lines in the block.</param>
01632 /// <param name="out_image">If the block contains an image, this pointer will point to it.</param>
01633 /// <param name="out_a">If the block contains an image, the first element of the image's
01634 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01635 /// <param name="out_b">If the block contains an image, the second element of the image's
01636 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01637 /// <param name="out_c">If the block contains an image, the third element of the image's
01638 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01639 /// <param name="out_d">If the block contains an image, the fourth element of the image's
01640 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01641 /// <param name="out_e">If the block contains an image, the fifth element of the image's
01642 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01643 /// <param name="out_f">If the block contains an image, the sixth element of the image's
01644 // transformation matrix [ [ a b 0 ] [ c d 0 ] [ e f 1 ] ].</param>
01645 /// <param name="out_stroked">If the block contains vector graphics, whether the graphics is
01646 // stroked.</param>
01647 /// <param name="out_argb">If the block contains stroked vector graphics, the R component of the
01648 // stroke colour.</param>
01649 /// <param name="out_xs_len">If the block contains "grid" lines, the number of X grid lines.</param>
01650 /// <param name="out_ys_len">If the block contains "grid" lines, the number of Y grid lines.</param>
01651 /// <param name="out_down">If the block is a structural element block, a pointer to the structural
01652 // block contents.</param>
01653 /// <param name="out_index">If the block is a structural element block, the index of the block within
01654 // the current level of the tree.</param>
01655 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01656 // occurred.</returns>
01657 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]

```

```

01632     internal static extern int GetStructuredTextBlock(IntPtr ctx, IntPtr block, ref int out_type,
01633     ref float out_x0, ref float out_y0, ref float out_x1, ref float out_y1, ref int out_line_count, ref
01634     IntPtr out_image, ref float out_a, ref float out_b, ref float out_c, ref float out_d, ref float out_e,
01635     ref float out_f, ref byte out_stroked, ref uint out_argb, ref int out_xs_len, ref int out_ys_len, ref
01636     IntPtr out_down, ref int out_index);
01637 /// <summary>
01638 /// Get an array of structured text blocks from a structured text page.
01639 /// </summary>
01640 /// <param name="page">The structured text page from which the blocks should be extracted.</param>
01641 /// <param name="out_blocks">An array of pointers to the structured text blocks.</param>
01642 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01643     occurred.</returns>
01644     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01645     internal static extern int GetStructuredTextBlocks(IntPtr page, IntPtr out_blocks);
01646 /// <summary>
01647 /// Get a structured text representation of a display list.
01648 /// </summary>
01649 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01650 /// <param name="list">The display list whose structured text representation is sought.</param>
01651 /// <param name="out_page">The address of the structured text page.</param>
01652 /// <param name="out_stext_block_count">The number of structured text blocks in the page.</param>
01653 /// <param name="flags">An integer equivalent to <see cref="StructuredText.StructuredTextFlags"/>,
01654     specifying flags for the structured text creation.</param>
01655 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01656     occurred.</returns>
01657     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01658     internal static extern int GetStructuredTextPage(IntPtr ctx, IntPtr list, int flags, ref
01659     IntPtr out_page, ref int out_stext_block_count);
01660 /// <summary>
01661 /// Delegate defining a callback function that is invoked by the unmanaged MuPDF library to indicate
01662     OCR progress.
01663 /// </summary>
01664 /// <param name="progress">The current progress, ranging from 0 to 100.</param>
01665 /// <returns>This function should return 0 to indicate that the OCR process should continue, or 1 to
01666     indicate that it should be stopped.</returns>
01667     internal delegate int ProgressCallback(int progress);
01668 /// <summary>
01669 /// Get a structured text representation of a display list, using the Tesseract OCR engine.
01670 /// </summary>
01671 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01672 /// <param name="list">The display list whose structured text representation is sought.</param>
01673 /// <param name="flags">An integer equivalent to <see cref="StructuredText.StructuredTextFlags"/>,
01674     specifying flags for the structured text creation.</param>
01675 /// <param name="out_page">The address of the structured text page.</param>
01676 /// <param name="out_stext_block_count">The number of structured text blocks in the page.</param>
01677 /// <param name="zoom">How much the specified region should be scaled when rendering. This determines
01678     the size in pixels of the image that is passed to Tesseract.</param>
01679 /// <param name="x0">The left coordinate in page units of the region of the display list that should
01680     be analysed.</param>
01681 /// <param name="y0">The top coordinate in page units of the region of the display list that should be
01682     analysed.</param>
01683 /// <param name="x1">The right coordinate in page units of the region of the display list that should
01684     be analysed.</param>
01685 /// <param name="y1">The bottom coordinate in page units of the region of the display list that should
01686     be analysed.</param>
01687 /// <param name="prefix">A string value that will be used as an argument for the <c>putenv</c>
01688     function. If this is <see langword="null"/>, the <c>putenv</c> function is not invoked. Usually used
01689     to set the value of the <c>TESSDATA_PREFIX</c> environment variable.</param>
01690 /// <param name="language">The name of the language model file to use for the OCR.</param>
01691 /// <param name="callback">A progress callback function. This function will be called with an integer
01692     parameter ranging from 0 to 100 to indicate OCR progress, and should return 0 to continue or 1 to
01693     abort the OCR process.</param>
01694 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01695     occurred.</returns>
01696     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01697     internal static extern int GetStructuredTextPageWithOCR(IntPtr ctx, IntPtr list, int flags,
01698     ref IntPtr out_page, ref int out_stext_block_count, float zoom, float x0, float y0, float x1, float
01699     y1, string prefix, string language, [MarshalAs(UnmanagedType.FunctionPtr)] ProgressCallback callback);
01700 /// <summary>
01701 /// Free a native structured text page and its associated resources.
01702 /// </summary>
01703 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01704 /// <param name="page">The structured text page to free.</param>
01705 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01706     occurred.</returns>
01707     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01708     internal static extern int DisposeStructuredTextPage(IntPtr ctx, IntPtr page);
01709 /// <summary>
01710 /// Redirect the standard output and standard error to named pipes with the specified names. On
01711     Windows, these are actually named pipes; on Linux and macOS, these are Unix sockets (matching the
01712     behaviour of System.IO.Pipes). Note that this has side-effects.

```

```

01693 /// </summary>
01694 /// <param name="stdoutFD">When the method returns, this variable will contain the file descriptor
01695     // corresponding to the "real" stdout.</param>
01696 /// <param name="stderrFD">When the method returns, this variable will contain the file descriptor
01697     // corresponding to the "real" stderr.</param>
01698 /// <param name="stdoutPipeName">The name of the pipe where stdout will be redirected. On Windows,
01699     // this should be of the form "\\.\pipe\xxx", while on Linux and macOS it should be an absolute file path
01700     // (maximum length 107/108 characters).</param>
01701 /// <param name="stderrPipeName">The name of the pipe where stderr will be redirected. On Windows,
01702     // this should be of the form "\\.\pipe\xxx", while on Linux and macOS it should be an absolute file path
01703     // (maximum length 107/108 characters).</param>
01704     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01705         internal static extern void RedirectOutput(out int stdoutFD, out int stderrFD, string
01706             stdoutPipeName, string stderrPipeName);
01707
01708 /// <summary>
01709     // Write the specified <paramref name="text"/> to a file descriptor. Use 1 for stdout and 2 for
01710     // stderr (which may have been redirected).
01711 /// </summary>
01712 /// <param name="fileDescriptor">The file descriptor on which to write.</param>
01713 /// <param name="text">The text to write.</param>
01714 /// <param name="length">The length of the text to write (i.e., text.Length).</param>
01715     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01716         internal static extern void WriteToFileDescriptor(int fileDescriptor, string text, int
01717             length);
01718
01719 /// <summary>
01720     // Resets the standard output and standard error (or redirect them to the specified file descriptors,
01721     // theoretically). Use with the <paramref name="stdoutFD"/> and <paramref name="stderrFD"/> returned by
01722     // <see cref="RedirectOutput"/> to undo what it did.
01723 /// </summary>
01724 /// <param name="stdoutFD">The file descriptor corresponding to the "real" stdout.</param>
01725 /// <param name="stderrFD">The file descriptor corresponding to the "real" stderr.</param>
01726     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01727         internal static extern void ResetOutput(int stdoutFD, int stderrFD);
01728
01729 /// <summary>
01730     // Unlocks a document with a password.
01731 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01732 /// <param name="doc">The document that needs to be unlocked.</param>
01733 /// <param name="password">The password to unlock the document.</param>
01734 /// <returns>0 if the document could not be unlocked, 1 if the document did not require unlocking in
01735     // the first place, 2 if the document was unlocked using the user password and 4 if the document was
01736     // unlocked using the owner password.</returns>
01737     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01738         internal static extern int UnlockWithPassword(IntPtr ctx, IntPtr doc, string password);
01739
01740 /// <summary>
01741     // Checks whether a password is required to open the document.
01742 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01743 /// <param name="doc">The document that needs to be checked.</param>
01744 /// <returns>0 if a password is not needed, 1 if a password is needed.</returns>
01745     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01746         internal static extern int CheckIfPasswordNeeded(IntPtr ctx, IntPtr doc);
01747
01748 /// <summary>
01749     // Returns the current permissions for the document. Note that these are not actually enforced.
01750 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01751 /// <param name="doc">The document whose permissions need to be checked.</param>
01752 /// <returns>An integer with bit 0 set if the document can be printed, bit 1 set if it can be copied,
01753     // bit 2 set if it can be edited, and bit 3 set if it can be annotated.</returns>
01754     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01755         internal static extern int GetPermissions(IntPtr ctx, IntPtr doc);
01756
01757 /// <summary>
01758     // Loads the document outline (table of contents).
01759 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01760 /// <param name="doc">The document whose outline should be loaded.</param>
01761     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01762         internal static extern IntPtr LoadOutline(IntPtr ctx, IntPtr doc);
01763
01764 /// <summary>
01765     // Frees memory allocated by a document outline (table of contents).
01766 /// <param name="outline">The document outline whose allocated memory should be released.</param>
01767     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01768         internal static extern void DisposeOutline(IntPtr ctx, IntPtr outline);
01769
01770 /// <summary>
01771     // Gathers metadata about an image.
01772 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>

```

```

01766 /// <param name="image">A pointer to the image.</param>
01767 /// <param name="out_w">When this method returns, this variable will contain the width of the
01768 /// image.</param>
01769 /// <param name="out_h">When this method returns, this variable will contain the height of the
01770 /// image.</param>
01771 /// <param name="out_xres">When this method returns, this variable will contain the horizontal
01772 /// resolution of the image.</param>
01773 /// <param name="out_yres">When this method returns, this variable will contain the vertical
01774 /// resolution of the image.</param>
01775 /// <param name="out_orientation">When this method returns, this variable will contain the orientation
01776 /// of the image.</param>
01777 /// <param name="out_colorspace">When this method returns, this variable will contain a pointer to the
01778 /// colour space of the image.</param>
01779 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01780 /// occurred.</returns>
01781 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01782     internal static extern int GetImageMetadata(IntPtr ctx, IntPtr image, ref int out_w, ref int
01783     out_h, ref int out_xres, ref int out_yres, ref byte out_orientation, ref IntPtr out_colorspace);
01784
01785 /// <summary>
01786 /// Write an image onto a stream in the specified format.
01787 /// </summary>
01788 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01789 /// <param name="image">A pointer to the image.</param>
01790 /// <param name="output_format">The output format.</param>
01791 /// <param name="quality">For JPEG output, the quality value.</param>
01792 /// <param name="out_buffer">The address of the buffer on which the data has been written (only useful
01793 /// for disposing the buffer later).</param>
01794 /// <param name="out_data">The address of the byte array where the data has been actually
01795 /// written.</param>
01796 /// <param name="out_length">The length in bytes of the image data.</param>
01797 /// <param name="convert_to_rgb">If this is 1, the image is converted to the RGB colour space before
01798 /// being exported.</param>
01799 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01800 /// occurred.</returns>
01801 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01802     internal static extern int WriteRasterImage(IntPtr ctx, IntPtr image, int output_format, int
01803     quality, ref IntPtr out_buffer, ref IntPtr out_data, ref ulong out_length, int convert_to_rgb);
01804
01805 /// <summary>
01806 /// Save an image to a file in the specified format.
01807 /// </summary>
01808 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01809 /// <param name="image">A pointer to the image.</param>
01810 /// <param name="output_format">The output format.</param>
01811 /// <param name="quality">For JPEG output, the quality value.</param>
01812 /// <param name="file_name">The name of the output file.</param>
01813 /// <param name="convert_to_rgb">If this is 1, the image is converted to the RGB colour space before
01814 /// being exported.</param>
01815 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01816 /// occurred.</returns>
01817 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01818     internal static extern int SaveRasterImage(IntPtr ctx, IntPtr image, string file_name, int
01819     output_format, int quality, int convert_to_rgb);
01820
01821 /// <summary>
01822 /// Release the resources associated with a pixmap.
01823 /// </summary>
01824 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01825 /// <param name="pixmap">The pixmap to be released.</param>
01826 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01827 /// occurred.</returns>
01828 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01829     internal static extern void DisposePixmap(IntPtr ctx, IntPtr pixmap);
01830
01831 /// <summary>
01832 /// Load image data from an image onto a pixmap, after converting it to the specified pixel format.
01833 /// </summary>
01834 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01835 /// <param name="image">A pointer to the image.</param>
01836 /// <param name="color_format">The <see cref="PixelFormats"/> to which the image should be
01837 /// converted.</param>
01838 /// <param name="out_pixmap">When this method returns, this variable will contain a pointer to the
01839 /// pixmap.</param>
01840 /// <param name="out_samples">When this method returns, this variable will contain a pointer to the
01841 /// image data.</param>
01842 /// <param name="count">The size in bytes of the image data.</param>
01843 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01844 /// occurred.</returns>
01845 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01846     internal static extern int LoadPixmapRGB(IntPtr ctx, IntPtr image, int color_format, ref
01847     IntPtr out_pixmap, ref IntPtr out_samples, ref int count);
01848
01849 /// <summary>
01850 /// Load image data from an image onto a pixmap.
01851 /// </summary>
01852 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01853 /// <param name="image">A pointer to the image.</param>

```

```

01832 /// <param name="out_pixmap">When this method returns, this variable will contain a pointer to the
01833     pixmap.</param>
01834 /// <param name="out_samples">When this method returns, this variable will contain a pointer to the
01835     image data.</param>
01836 /// <param name="count">The size in bytes of the image data.</param>
01837 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01838     occurred.</returns>
01839     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01840     internal static extern int LoadPixmap(IntPtr ctx, IntPtr image, ref IntPtr out_pixmap, ref
01841     IntPtr out_samples, ref int count);
01842
01843 /// <summary>
01844     Get the name of a colour space.
01845 /// </summary>
01846 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01847 /// <param name="cs">A pointer to the colour space.</param>
01848 /// <param name="length">The length of the name.</param>
01849 /// <param name="out_name">A pointer to a byte array where the name will be copied.</param>
01850 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01851     occurred.</returns>
01852     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01853     internal static extern int GetColorSpaceName(IntPtr ctx, IntPtr cs, int length, IntPtr
01854     out_name);
01855
01856 /// <summary>
01857     Get information about a colour space.
01858 /// </summary>
01859 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01860 /// <param name="cs">A pointer to the colour space.</param>
01861 /// <param name="out_cs_type">The type of colour space (equivalent to <see
01862     cref="ColorSpaceType"/>).</param>
01863 /// <param name="out_name_len">When this method returns, this variable will contain the length of the
01864     name of the colour space.</param>
01865 /// <param name="out_base_cs">When this method returns, this variable will contain a pointer to the
01866     base colour space (for indexed colour spaces) or to the alternate colour space (for separations colour
01867     spaces).</param>
01868 /// <param name="out_lookup_size">When this method returns, this variable will contain the number of
01869     colours used by the image (for indexed colour spaces) or the number of colourants (for separations
01870     colour spaces).</param>
01871 /// <param name="out_lookup_table">When this method returns, this variable will contain a pointer to
01872     the colour lookup table (for indexed colour spaces).</param>
01873 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01874     occurred.</returns>
01875     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01876     internal static extern int GetColorSpaceData(IntPtr ctx, IntPtr cs, ref int out_cs_type, ref
01877     int out_name_len, ref IntPtr out_base_cs, ref int out_lookup_size, ref IntPtr out_lookup_table);
01878
01879 /// <summary>
01880     Get the length of the name of a colourant.
01881 /// </summary>
01882 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01883 /// <param name="cs">A pointer to the colour space.</param>
01884 /// <param name="n">The index of the colourant.</param>
01885 /// <param name="out_name_length">When this method returns, this variable will contain the length of
01886     the name of the colourant.</param>
01887 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01888     occurred.</returns>
01889     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01890     internal static extern int GetColorantNameLength(IntPtr ctx, IntPtr cs, int n, ref int
01891     out_name_length);
01892
01893 /// <summary>
01894     Get the name of a colourant.
01895 /// </summary>
01896 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01897 /// <param name="cs">A pointer to the colour space.</param>
01898 /// <param name="n">The index of the colourant.</param>
01899 /// <param name="length">The length of the name of the colourant.</param>
01900 /// <param name="out_name">A pointer to a byte array where the name will be copied.</param>
01901 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01902     occurred.</returns>
01903     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01904     internal static extern int GetColorantName(IntPtr ctx, IntPtr cs, int n, int length, IntPtr
01905     out_name);
01906
01907 /// <summary>
01908     Get the Type3 procs for a font.
01909 /// </summary>
01910 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01911 /// <param name="font">The font.</param>
01912 /// <param name="out_t3_procs">When this method returns, this variable will contain a pointer to the
01913     Type3 procs for the font (if any).</param>
01914 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
01915     occurred.</returns>
01916     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01917     internal static extern int GetT3Procs(IntPtr ctx, IntPtr font, ref IntPtr out_t3_procs);
01918
01919

```

```

01897 /// <summary>
01898 /// Get the FreeType FT_Face handle for a font.
01899 /// </summary>
01900 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01901 /// <param name="font">The font.</param>
01902 /// <param name="out_handle">When this method returns, this variable will contain a pointer to the
FreeType FT_Face handle for the font (if any).</param>
01903 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
occurred.</returns>
01904     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01905     internal static extern int GetFTHandle(IntPtr ctx, IntPtr font, ref IntPtr out_handle);
01906
01907 /// <summary>
01908 /// Get the name of a font.
01909 /// </summary>
01910 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01911 /// <param name="font">The font.</param>
01912 /// <param name="length">The length of the name of the font.</param>
01913 /// <param name="out_name">A pointer to a byte array where the name will be copied.</param>
01914 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
occurred.</returns>
01915     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01916     internal static extern int GetFontName(IntPtr ctx, IntPtr font, int length, IntPtr out_name);
01917
01918 /// <summary>
01919 /// Get information about a font.
01920 /// </summary>
01921 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01922 /// <param name="font">The font.</param>
01923 /// <param name="out_font_name_length">When this method returns, this variable will hold the number of
characters in the font's name.</param>
01924 /// <param name="out_bold">When this method returns, this variable will be 1 if the font is
bold.</param>
01925 /// <param name="out_italic">When this method returns, this variable will be 1 if the font is
italic.</param>
01926 /// <param name="out_serif">When this method returns, this variable will be 1 if the font has serifs
(not very reliable).</param>
01927 /// <param name="out_monospaced">When this method returns, this variable will be 1 if the font is
monospaced.</param>
01928 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
occurred.</returns>
01929     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01930     internal static extern int GetFontMetadata(IntPtr ctx, IntPtr font, ref int
out_font_name_length, ref int out_bold, ref int out_italic, ref int out_serif, ref int
out_monospaced);
01931
01932 /// <summary>
01933 /// Release the resources associated with the specified font.
01934 /// </summary>
01935 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01936 /// <param name="font">The font.</param>
01937     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01938     internal static extern void DisposeFont(IntPtr ctx, IntPtr font);
01939
01940 /// <summary>
01941 /// Release the resources associated with the specified image.
01942 /// </summary>
01943 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01944 /// <param name="image">The image.</param>
01945     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01946     internal static extern void DisposeImage(IntPtr ctx, IntPtr image);
01947
01948 /// <summary>
01949 /// Release the resources associated with the specified colour space.
01950 /// </summary>
01951 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01952 /// <param name="cs">The colour space.</param>
01953     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01954     internal static extern void DisposeColorSpace(IntPtr ctx, IntPtr cs);
01955
01956 /// <summary>
01957 /// Set the state of an optional content group "UI" element.
01958 /// </summary>
01959 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01960 /// <param name="doc">The PDF document.</param>
01961 /// <param name="ui_index">The index of the optional content group "UI" element.</param>
01962 /// <param name="state">The state to set (<c>0</c> to uncheck the element, <c>1</c> to check it, or
<c>2</c> to toggle it).</param>
01963     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01964     internal static extern void SetOptionalContentGroupUIState(IntPtr ctx, IntPtr doc, int
ui_index, int state);
01965
01966 /// <summary>
01967 /// Get the state of an optional content group "UI" element.
01968 /// </summary>
01969 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01970 /// <param name="doc">The PDF document.</param>

```

```

01971 /// <param name="ui_index">The index of the optional content group "UI" element.</param>
01972 /// <returns>If the optional content group is disabled, this method will return <c>0</c>, otherwise a
01973     value different from <c>0</c>.</returns>
01974     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01975     internal static extern int ReadOptionalContentGroupUIState(IntPtr ctx, IntPtr doc, int
01976     ui_index);
01975
01976 /// <summary>
01977 /// Get information about the optional content group "UI" elements.
01978 /// </summary>
01979 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01980 /// <param name="doc">The PDF document.</param>
01981 /// <param name="count">The number of optional content group "UI" elements.</param>
01982 /// <param name="out_labels">A pointer to an array that can hold <paramref name="count"/> <see
01983     langword="string"/>s, whose length can be obtained using <see
01984     cref="ReadOptionalContentGroupUILabelLengths"/></param>
01985 /// <param name="out_depths">A pointer to an array that can hold <paramref name="count"/> <see
01986     langword="int"/>s, which will be filled when this method returns.</param>
01987 /// <param name="out_types">A pointer to an array that can hold <paramref name="count"/> <see
01988     langword="int"/>s, which will be filled when this method returns.</param>
01989 /// <param name="out_locked">A pointer to an array that can hold <paramref name="count"/> <see
01990     langword="int"/>s, which will be filled when this method returns.</param>
01991     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
01992     internal static extern void ReadOptionalContentGroupUIs(IntPtr ctx, IntPtr doc, int count,
01993     IntPtr out_labels, IntPtr out_depths, IntPtr out_types, IntPtr out_locked);
01994
01995 /// <summary>
01996 /// Get the lengths of the optional content group "UI" labels.
01997 /// </summary>
01998 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
01999 /// <param name="doc">The PDF document.</param>
02000 /// <param name="count">The number of optional content group "UI" elements.</param>
02001 /// <param name="out_lengths">A pointer to an array that can hold <paramref name="count"/> <see
02002     langword="int"/>s, which will be filled when this method returns.</param>
02003     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02004     internal static extern void ReadOptionalContentGroupUILabelLengths(IntPtr ctx, IntPtr doc, int
02005     count, IntPtr out_lengths);
02006
02007 /// <summary>
02008 /// Get the number of optional content group "UI" elements.
02009 /// </summary>
02010 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02011 /// <param name="doc">The PDF document.</param>
02012 /// <returns>The number of optional content group "UI" elements.</returns>
02013     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02014     internal static extern int CountOptionalContentGroupConfigUI(IntPtr ctx, IntPtr doc);
02015
02016 /// <summary>
02017 /// Set the state of an optional content group.
02018 /// </summary>
02019 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02020 /// <param name="doc">The PDF document.</param>
02021 /// <param name="index">The index of the optional content group.</param>
02022 /// <param name="state">The state to set (<c>0</c> for disabled, any other value for enabled)</param>
02023     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02024     internal static extern void SetOptionalContentGroupState(IntPtr ctx, IntPtr doc, int index,
02025     int state);
02026
02027 /// <summary>
02028 /// Get the state of an optional content group.
02029 /// </summary>
02030 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02031 /// <param name="doc">The PDF document.</param>
02032 /// <param name="index">The index of the optional content group.</param>
02033 /// <returns>If the optional content group is disabled, this method will return <c>0</c>, otherwise a
02034     value different from <c>0</c>.</returns>
02035     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02036     internal static extern int GetOptionalContentGroupState(IntPtr ctx, IntPtr doc, int index);
02037
02038 /// <summary>
02039 /// Get the optional content group names.
02040 /// </summary>
02041 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02042 /// <param name="doc">The PDF document.</param>

```

```
02043 /// <param name="count">The number of optional content groups.</param>
02044 /// <param name="out_lengths">A pointer to an array that can hold <paramref name="count"/> <see
02045     cref="int"/>s, which will be filled when this method returns.</param>
02046     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02047     internal static extern void GetOptionalContentGroupNameLengths(IntPtr ctx, IntPtr doc, int
02048         count, IntPtr out_lengths);
02049
02050     /// <summary>
02051     /// Get the number of optional content groups defined in the document.
02052     /// </summary>
02053     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02054     /// <param name="doc">The PDF document.</param>
02055     /// <returns>The number of optional content groups defined in the document.</returns>
02056     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02057     internal static extern int CountOptionalContentGroups(IntPtr ctx, IntPtr doc);
02058
02059     /// <summary>
02060     /// Activate an alternative optional content group configuration.
02061     /// </summary>
02062     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02063     /// <param name="doc">The PDF document.</param>
02064     /// <param name="configuration_index">The index of the optional content group configuration to
02065         activate.</param>
02066     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02067     internal static extern void EnableOCGConfig(IntPtr ctx, IntPtr doc, int configuration_index);
02068
02069     /// <summary>
02070     /// Activate the default optional content group configuration.
02071     /// </summary>
02072     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02073     /// <param name="doc">The PDF document.</param>
02074     /// <param name="configuration_index">The index of the alternative optional content group
02075         configuration.</param>
02076     /// <param name="name_length">The length of the name of optional content group configuration.</param>
02077     /// <param name="creator_length">The length of the creator of optional content group
02078         configuration.</param>
02079     /// <param name="out_name">A pointer to a byte array where the name will be copied.</param>
02080     /// <param name="out_creator">A pointer to a byte array where the creator will be copied.</param>
02081     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02082     internal static extern void ReadOCGConfig(IntPtr ctx, IntPtr doc, int configuration_index, int
02083         name_length, int creator_length, IntPtr out_name, IntPtr out_creator);
02084
02085     /// <summary>
02086     /// Get the length of the name and creator of an alternative optional content group configuration.
02087     /// </summary>
02088     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02089     /// <param name="doc">The PDF document.</param>
02090     /// <param name="configuration_index">The index of the alternative optional content group
02091         configuration.</param>
02092     /// <param name="out_name_length">When this method returns, this variable will contain the length of
02093         the name of the optional content group configuration.</param>
02094     /// <param name="out_creator_length">When this method returns, this variable will contain the length
02095         of the creator of the optional content group configuration.</param>
02096     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02097     internal static extern void ReadOCGConfigNameLength(IntPtr ctx, IntPtr doc, int
02098         configuration_index, ref int out_name_length, ref int out_creator_length);
02099
02100     /// <summary>
02101     /// Get the number of alternative optional content group configurations.
02102     /// </summary>
02103     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02104     /// <param name="doc">The PDF document.</param>
02105     /// <returns>The number of alternative optional content group configurations.</returns>
02106     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02107     internal static extern int CountAlternativeOCGConfigs(IntPtr ctx, IntPtr doc);
02108
02109     /// <summary>
02110     /// Get the name and creator of the default optional content group configuration.
02111     /// </summary>
02112     /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02113     /// <param name="name_length">The length of the name of the default optional content group
02114         configuration.</param>
02115     /// <param name="creator_length">The length of the creator of the default optional content group
02116         configuration.</param>
02117     /// <param name="out_name">A pointer to a byte array where the name will be copied.</param>
02118     /// <param name="out_creator">A pointer to a byte array where the creator will be copied.</param>
02119     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02120     internal static extern void ReadDefaultOCGConfig(IntPtr ctx, IntPtr doc, int name_length, int
```

```

    creator_length, IntPtr out_name, IntPtr out_creator);
02118
02119 /// <summary>
02120 /// Get the length of the name and creator of the default optional content group configuration.
02121 /// </summary>
02122 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02123 /// <param name="doc">The PDF document.</param>
02124 /// <param name="out_name_length">When this method returns, this variable will contain the length of
02125 /// the name of the default optional content group configuration.</param>
02126 /// <param name="out_creator_length">When this method returns, this variable will contain the length
02127 /// of the creator of the default optional content group configuration.</param>
02128     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02129     internal static extern void ReadDefaultOCGConfigNameLength(IntPtr ctx, IntPtr doc, ref int
02129     out_name_length, ref int out_creator_length);
02130
02131 /// <summary>
02132 /// Cast a MuPDF document to a MuPDF PDF document.
02133 /// </summary>
02134 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02135 /// <param name="doc">The document to convert.</param>
02136 /// <param name="out_pdf_doc">The casted PDF document, or <see cref="IntPtr.Zero"/> if the document
02137 /// cannot be cast.</param>
02138 /// <returns>An integer equivalent to <see cref="ExitCodes"/> detailing whether any errors
02139 /// occurred.</returns>
02140     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02141     internal static extern int GetPDFDocument(IntPtr ctx, IntPtr doc, ref IntPtr out_pdf_doc);
02142
02143 /// <summary>
02144 /// Activate a SetOCGState link, thus hiding/showing some optional content groups.
02145 /// </summary>
02146 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02147 /// <param name="doc">The PDF document.</param>
02148 /// <param name="link">The link to activate.</param>
02149 /// <returns>The absolute page number.</returns>
02150 /// <summary>
02151 /// Get an absolute page number from a chapter number and page number.
02152 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02153 /// <param name="doc">The document.</param>
02154 /// <param name="chapter">The chapter number.</param>
02155 /// <param name="page">The page number within the chapter.</param>
02156 /// <returns>The absolute page number.</returns>
02157     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02158     internal static extern int GetPageNumber(IntPtr ctx, IntPtr doc, int chapter, int page);
02159
02160 /// <summary>
02161 /// Check whether a link is visible or not.
02162 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02163 /// <param name="usage">Usage string. Not sure if anything other than <c>"View"</c> works.</param>
02164 /// <param name="link">The link.</param>
02165 /// <returns><c>0</c> if the link is visible, any other value if it is hidden.</returns>
02166     [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02167     internal static extern int IsLinkHidden(IntPtr ctx, string usage, IntPtr link);
02168
02169 /// <summary>
02170 /// Gather information about a link.
02171 /// </summary>
02172 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02173 /// <param name="doc">The document that contains the link.</param>
02174 /// <param name="link">The link.</param>
02175 /// <param name="uri_length">The length of the link's Uri, as determined by <see
02176 /// cref="LoadLinks"/></param>
02177 /// <param name="isPDF">Set this to <c>1</c> if the document is a PDF document, or to <c>0</c>
02178 /// otherwise.</param>
02179 /// <param name="out_x0">When this method returns, this variable will contain the left coordinate of
02180 /// the link's active rectangle.</param>
02181 /// <param name="out_y0">When this method returns, this variable will contain the top coordinate of
02182 /// the link's active rectangle.</param>
02183 /// <param name="out_x1">When this method returns, this variable will contain the right coordinate of
02184 /// the link's active rectangle.</param>
02185 /// <param name="out_y1">When this method returns, this variable will contain the bottom coordinate of
02186 /// the link's active rectangle.</param>
02187 /// <param name="out_uri">A pointer to an array of <see cref="byte"/>s that will be filled with the
02188 /// link's Uri.</param>
02189 /// <param name="out_is_external">When this method returns, this variable will be <c>0</c> if the link
02190 /// is not an external link, and a different value otherwise.</param>
02191 /// <param name="out_is_setocgstate">When this method returns, this variable will be <c>0</c> if the
02192 /// link is not a SetOCGState link, and a different value otherwise.</param>
02193 /// <param name="out_dest_type">If the link is an internal link, when this method returns, this
02194 /// variable will contain the destination type of the internal link.</param>
02195 /// <param name="out_dest_x">If the link is an internal link, when this method returns, this variable
02196 /// will contain the X coordinate of the internal link target.</param>
02197 /// <param name="out_dest_y">If the link is an internal link, when this method returns, this variable
02198 /// will contain the Y coordinate of the internal link target.</param>

```

```

02187 /// <param name="out_dest_w">If the link is an internal link, when this method returns, this variable
02188 /// will contain the width of the internal link target.</param>
02189 /// <param name="out_dest_h">If the link is an internal link, when this method returns, this variable
02190 /// will contain the height of the internal link target.</param>
02191 /// <param name="out_dest_zoom">If the link is an internal link, when this method returns, this
02192 /// variable will contain the zoom of the internal link target.</param>
02193 /// <param name="out_dest_page">If the link is an internal link, when this method returns, this
02194 /// variable will contain the destination page of the internal link.</param>
02195 /// <param name="out_dest_chapter">If the link is an internal link, when this method returns, this
02196 /// variable will contain the destination chapter of the internal link.</param>
02197 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02198     internal static extern void LoadLink(IntPtr ctx, IntPtr doc, IntPtr link, int uri_length, int
02199     isPDF, ref float out_x0, ref float out_y0, ref float out_x1, ref float out_y1, IntPtr out_uri, ref int
02200     out_is_external, ref int out_is_setocgstate, ref int out_dest_type, ref float out_dest_x, ref float
02201     out_dest_y, ref float out_dest_w, ref float out_dest_h, ref float out_dest_zoom, ref int
02202     out_dest_page, ref int out_dest_chapter);
02203 /// <summary>
02204 /// Free resources associated with a link collection.
02205 /// </summary>
02206 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02207 /// <param name="firstLink">The first link in the collection.</param>
02208 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02209     internal static extern void DisposeLinks(IntPtr ctx, IntPtr firstLink);
02210
02211 /// <summary>
02212 /// Load links from a page.
02213 /// </summary>
02214 /// <param name="firstLink">The first link in the page.</param>
02215 /// <param name="out_links">A pointer to an array of <see cref="IntPtr"/>s that will be filled by this
02216 /// method.</param>
02217 /// <param name="out_uri_lengths">A pointer to an array of <see cref="int"/>/s that will be filled by
02218 /// this method (these represent the lengths of the links' Uris).</param>
02219 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02220     internal static extern void LoadLinks(IntPtr firstLink, IntPtr out_links, IntPtr
02221     out_uri_lengths);
02222 }
02223
02224 /// <summary>
02225 /// Count the number of links in the page and return a pointer to the first link.
02226 /// </summary>
02227 /// <param name="ctx">A context to hold the exception stack and the cached resources.</param>
02228 /// <param name="page">A pointer to the page from which links should be extracted.</param>
02229 /// <param name="out_firstLink">When this method returns, this variable will contain a pointer to the
02230 /// first link in the page.</param>
02231 /// <returns>The number of links contained in the page.</returns>
02232 [DllImport("MuPDFWrapper", CallingConvention = CallingConvention.Cdecl)]
02233     internal static extern int CountLinks(IntPtr ctx, IntPtr page, ref IntPtr out_firstLink);
02234 }
02235 }
```

8.5 MuPDFColorSpace.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2024 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Text;
00020
00021 namespace MuPDFCore
00022 {
00023
00024 /// <summary>
00025 /// Colour space used by the image.
00026 /// </summary>
00027 public enum ColorSpaceType
00028 {
00029 /// <summary>
00030 /// Not specified.
00031 /// </summary>

```

```
00032     None = 0,
00033
00034 /// <summary>
00035 /// Grayscale. Each pixel is stored as a single byte.
00036 /// </summary>
00037     Gray = 1,
00038
00039 /// <summary>
00040 /// RGB colour space. Each pixel is stored as three bytes representing the R, G, and B components of
00041 /// the colour in this order.
00042     RGB = 2,
00043
00044 /// <summary>
00045 /// BGR colour space. Each pixel is stored as three bytes representing the B, G, and R components of
00046 /// the colour in this order.
00047     BGR = 3,
00048
00049 /// <summary>
00050 /// CMYK colour space. Each pixel is stored as four bytes representing the C, M, Y, and K components
00051 /// of the colour in this order.
00052     CMYK = 4,
00053
00054 /// <summary>
00055 /// Lab colour space. Each pixel is stored as three bytes representing the L*, a, and b components of
00056 /// the colour in this order.
00057     Lab = 5,
00058
00059 /// <summary>
00060 /// Indexed colour space. Each pixel is stored as a single byte mapping to an entry in the colour
00061 /// space palette.
00062     Indexed = 6,
00063
00064 /// <summary>
00065 /// Separation colour space.
00066 /// </summary>
00067     Separation = 7
00068 }
00069
00070 /// <summary>
00071 /// Represents a colour space.
00072 /// </summary>
00073     public abstract class MuPDFColorSpace
00074     {
00075         /// <summary>
00076         /// The colour space type.
00077         /// </summary>
00078         public abstract ColorSpaceType Type { get; }
00079
00080 /// <summary>
00081 /// The number of bytes necessary to represent a single pixel in this colour space.
00082 /// </summary>
00083     public abstract int NumBytes { get; }
00084
00085 /// <summary>
00086 /// The name of the colour space.
00087 /// </summary>
00088     public virtual string Name { get; }
00089
00090 /// <summary>
00091 /// Create a new <see cref="MuPDFColorSpace"/> base instance.
00092 /// </summary>
00093 /// <param name="name">The name of the colour space.</param>
00094     protected internal MuPDFColorSpace(string name)
00095     {
00096         Name = name;
00097     }
00098
00099 /// <summary>
00100 /// The final colour space in which all pixels will eventually be represented.
00101 /// </summary>
00102     public virtual MuPDFColorSpace RootColorSpace
00103     {
00104         get
00105         {
00106             if (this is IndexedMuPDFColorSpace indexed)
00107             {
00108                 return indexed.BaseColorSpace.RootColorSpace;
00109             }
00110             else
00111             {
00112                 return this;
00113             }
00114         }
00115     }
00116 }
```

```

00114         }
00115     }
00116
00117     internal static unsafe MuPDFColorSpace Create(IntPtr nativeContext, IntPtr nativePointer)
00118     {
00119         int csType = 0;
00120         IntPtr baseColorSpace = IntPtr.Zero;
00121         IntPtr lookupPointer = IntPtr.Zero;
00122         int lookupLength = 0;
00123         int nameLength = 0;
00124
00125         ExitCodes result = (ExitCodes)NativeMethods.GetColorSpaceData(nativeContext,
00126             nativePointer, ref csType, ref nameLength, ref baseColorSpace, ref lookupLength, ref lookupPointer);
00127
00128         switch (result)
00129         {
00130             case ExitCodes.EXIT_SUCCESS:
00131                 break;
00132             case ExitCodes.ERR_COLORSPACE_METADATA:
00133                 throw new MuPDFException("An error occurred while retrieving colour space
00134             information!", result);
00135             default:
00136                 throw new MuPDFException("Unknown error!", result);
00137         }
00138
00139         byte[] nameArray = new byte[nameLength];
00140
00141         fixed (byte* nameArrayPtr = nameArray)
00142         {
00143             result = (ExitCodes)NativeMethods.GetColorSpaceName(nativeContext, nativePointer,
00144             nameLength, (IntPtr)nameArrayPtr);
00145
00146             switch (result)
00147             {
00148                 case ExitCodes.EXIT_SUCCESS:
00149                     break;
00150                 case ExitCodes.ERR_COLORSPACE_METADATA:
00151                     throw new MuPDFException("An error occurred while retrieving colour space
00152             information!", result);
00153             default:
00154                 throw new MuPDFException("Unknown error!", result);
00155         }
00156
00157         string name = Encoding.UTF8.GetString(nameArray);
00158
00159         ColorSpaceType type = (ColorSpaceType)csType;
00160
00161         switch (type)
00162         {
00163             case ColorSpaceType.None:
00164                 return null;
00165             case ColorSpaceType.Gray:
00166                 return new GrayscaleMuPDFColorSpace(name);
00167             case ColorSpaceType.RGB:
00168                 return new RGBMuPDFColorSpace(name);
00169             case ColorSpaceType.BGR:
00170                 return new BGRMuPDFColorSpace(name);
00171             case ColorSpaceType.CMYK:
00172                 return new CMYKMuPDFColorSpace(name);
00173             case ColorSpaceType.Lab:
00174                 return new LabMuPDFColorSpace(name);
00175             case ColorSpaceType.Indexed:
00176                 {
00177                     MuPDFColorSpace baseMuPDFColorSpace = Create(nativeContext, baseColorSpace);
00178                     byte[] lookupTable = new byte[lookupLength * baseMuPDFColorSpace.NumBytes];
00179
00180                     fixed (byte* lookupTablePtr = lookupTable)
00181                     {
00182                         Buffer.MemoryCopy((byte*)lookupPointer, lookupTablePtr, lookupLength *
00183                             baseMuPDFColorSpace.NumBytes, lookupLength * baseMuPDFColorSpace.NumBytes);
00184                     }
00185
00186                     return new IndexedMuPDFColorSpace(lookupTable, baseMuPDFColorSpace, name);
00187                 }
00188             case ColorSpaceType.Separation:
00189                 {
00190                     string[] colorantNames = new string[lookupLength];
00191
00192                     for (int i = 0; i < lookupLength; i++)
00193                     {
00194                         int colorantNameLength = 0;
00195
00196                         result = (ExitCodes)NativeMethods.GetColorantNameLength(nativeContext,
00197                             nativePointer, i, ref colorantNameLength);
00198                         switch (result)
00199                         {

```

```

00195             case ExitCodes.EXIT_SUCCESS:
00196                 break;
00197             case ExitCodes.ERR_COLORSPACE_METADATA:
00198                 throw new MuPDFException("An error occurred while retrieving
00199                 colorant names!", result);
00200         }
00201     }
00202 
00203     byte[] colorantNameArray = new byte[colorantNameLength];
00204 
00205     fixed (byte* colorantNamePtr = colorantNameArray)
00206     {
00207         result = (ExitCodes)NativeMethods.GetColorantName(nativeContext,
00208         nativePointer, i, colorantNameLength, (IntPtr)colorantNamePtr);
00209     }
00210 
00211     switch (result)
00212     {
00213         case ExitCodes.EXIT_SUCCESS:
00214             break;
00215         case ExitCodes.ERR_COLORSPACE_METADATA:
00216             throw new MuPDFException("An error occurred while retrieving
00217                 colorant names!", result);
00218         default:
00219             throw new MuPDFException("Unknown error!", result);
00220     }
00221 
00222     colorantNames[i] = Encoding.UTF8.GetString(colorantNameArray);
00223 
00224     MuPDFColorSpace alternateColorSpace = MuPDFColorSpace.Create(nativeContext,
00225         baseColorSpace);
00226 
00227         return new SeparationColorSpace(name, colorantNames, alternateColorSpace);
00228     }
00229     default:
00230         throw new MuPDFException("Unknown colour space type!", ExitCodes.UNKNOWN_ERROR);
00231     }
00232 /// <inheritdoc>
00233     public override string ToString() => this.Name;
00234 }
00235 
00236 /// <summary>
00237 /// Represents a colour space where each pixel is stored as a single byte.
00238 /// </summary>
00239 public class GrayscaleMuPDFColorSpace : MuPDFColorSpace
00240 {
00241 /// <inheritdoc>
00242     public override ColorSpaceType Type => ColorSpaceType.Gray;
00243 
00244 /// <inheritdoc>
00245     public override int NumBytes => 1;
00246 
00247     internal GrayscaleMuPDFColorSpace(string name) : base(name)
00248     {
00249     }
00250 
00251 }
00252 
00253 /// <summary>
00254 /// Represents a colour space where each pixel is stored as three bytes encoding the R, G, and B
00255 /// components.
00256 /// </summary>
00257 public class RGBMuPDFColorSpace : MuPDFColorSpace
00258 /// <inheritdoc>
00259     public override ColorSpaceType Type => ColorSpaceType.RGB;
00260 
00261 /// <inheritdoc>
00262     public override int NumBytes => 3;
00263 
00264     internal RGBMuPDFColorSpace(string name) : base(name)
00265     {
00266     }
00267 
00268 }
00269 
00270 /// <summary>
00271 /// Represents a colour space where each pixel is stored as three bytes encoding the B, G, and R
00272 /// components.
00273 /// </summary>
00274 public class BGRMuPDFColorSpace : MuPDFColorSpace
00275 /// <inheritdoc>

```

```
00276     public override ColorSpaceType Type => ColorSpaceType.BGR;
00277
00278 /// <inheritdoc/>
00279     public override int NumBytes => 3;
00280
00281     internal BGRMuPDFColorSpace(string name) : base(name)
00282     {
00283
00284     }
00285 }
00286
00287 /// <summary>
00288 /// Represents a colour space where each pixel is stored as three bytes encoding the L*, a, and b
00289 /// components.
00290 /// </summary>
00291     public class LabMuPDFColorSpace : MuPDFColorSpace
00292     {
00293         public override ColorSpaceType Type => ColorSpaceType.Lab;
00294
00295 /// <inheritdoc/>
00296         public override int NumBytes => 3;
00297
00298         internal LabMuPDFColorSpace(string name) : base(name)
00299         {
00300
00301         }
00302     }
00303
00304 /// <summary>
00305 /// Represents a colour space where each pixel is stored as four bytes encoding the C, M, Y, and K
00306 /// components.
00307 /// </summary>
00308     public class CMYKMuPDFColorSpace : MuPDFColorSpace
00309     {
00310         public override ColorSpaceType Type => ColorSpaceType.CMYK;
00311
00312 /// <inheritdoc/>
00313         public override int NumBytes => 4;
00314
00315         internal CMYKMuPDFColorSpace(string name) : base(name)
00316         {
00317
00318         }
00319     }
00320
00321 /// <summary>
00322 /// Represents a colour space where each pixel is stored as a single byte mapping the pixel to a
00323 /// colour in the lookup table.
00324 /// </summary>
00325     public class IndexedMuPDFColorSpace : MuPDFColorSpace
00326     {
00327         public override ColorSpaceType Type => ColorSpaceType.Indexed;
00328
00329 /// <inheritdoc/>
00330         public override int NumBytes => 1;
00331
00332 /// <summary>
00333 /// The lookup table containing the colour index.
00334 /// </summary>
00335         public byte[] LookupTable { get; }
00336
00337 /// <summary>
00338 /// The colour space in which the colours in the <see cref="LookupTable"/> are expressed.
00339 /// </summary>
00340         public MuPDFColorSpace BaseColorSpace { get; }
00341
00342         internal IndexedMuPDFColorSpace(byte[] lookupTable, MuPDFColorSpace baseColorSpace, string
00343             name) : base(name)
00344         {
00345             this.LookupTable = lookupTable;
00346             this.BaseColorSpace = baseColorSpace;
00347         }
00348
00349 /// <summary>
00350 /// Represents a separation colour space.
00351 /// </summary>
00352     public class SeparationColorSpace : MuPDFColorSpace
00353     {
00354         public override ColorSpaceType Type => ColorSpaceType.Separation;
00355
00356         public override int NumBytes => 1;
```

```

00359
00360 /// <summary>
00361 /// Names of the separation colorants.
00362 /// </summary>
00363     public string[] ColorantNames { get; }
00364
00365 /// <summary>
00366 /// Alternate colour space.
00367 /// </summary>
00368     public MuPDFColorSpace AlternateColorSpace { get; }
00369
00370     internal SeparationColorSpace(string name, string[] colorantNames, MuPDFColorSpace
00371         alternateColorSpace) : base(name)
00372     {
00373         this.ColorantNames = colorantNames;
00374         this.AlternateColorSpace = alternateColorSpace;
00375     }
00376 }
```

8.6 MuPDFContext.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections.Generic;
00020
00021 namespace MuPDFCore
00022 {
00023 /// <summary>
00024 /// A wrapper around a MuPDF context object, which contains the exception stack and the resource cache
00025 /// store.
00026 /// </summary>
00027     public class MuPDFContext : IDisposable
00028     {
00029     /// <summary>
00030     /// Parent context for cloned contexts.
00031     /// </summary>
00032         private MuPDFContext ParentContext { get; } = null;
00033
00034     /// <summary>
00035     /// A pointer to the native context object.
00036     /// </summary>
00037         internal readonly IntPtr NativeContext;
00038
00039     /// <summary>
00040     /// The current size in bytes of the resource cache store. Read-only.
00041     /// </summary>
00042         public long StoreSize
00043         {
00044             get
00045             {
00046                 return (long)NativeMethods.GetCurrentStoreSize(this.NativeContext);
00047             }
00048         }
00049
00050         private object fontCacheLock = new object();
00051         private Dictionary<IntPtr, (MuPDFFont, int)> fontCache = new Dictionary<IntPtr, (MuPDFFont,
00052             int)>();
00053
00054     /// <summary>
00055     /// Font cache dictionary.
00056     /// </summary>
00057         internal Dictionary<IntPtr, (MuPDFFont, int)> FontCache
00058         {
00059             get
00060             {
00061                 return this.ParentContext == null ? fontCache : this.ParentContext.FontCache;
00062             }
00063         }
00064     }
```

```
00060         }
00061     }
00062
00063 /// <summary>
00064 /// Font cache dictionary lock.
00065 /// </summary>
00066     internal object FontCacheLock
00067     {
00068         get
00069         {
00070             return this.ParentContext == null ? fontCacheLock :
00071                 this.ParentContext.FontCacheLock;
00072         }
00073
00074 /// <summary>
00075 /// The maximum size in bytes of the resource cache store. Read-only.
00076 /// </summary>
00077     public long StoreMaxSize
00078     {
00079         get
00080         {
00081             return (long)NativeMethods.GetMaxStoreSize(this.NativeContext);
00082         }
00083     }
00084
00085 /// <summary>
00086 /// Sets the current anti-aliasing level. Changing this value will affect both
00087 /// the <see cref="TextAntiAliasing"/> and the <see cref="GraphicsAntiAliasing"/>.
00088 /// </summary>
00089     public int AntiAliasing
00090     {
00091         set
00092         {
00093             if (value < 0 || value > 8)
00094             {
00095                 throw new ArgumentOutOfRangeException(nameof(value), value, "The anti-aliasing
level must range between 0 and 8 (inclusive).");
00096             }
00097
00098             NativeMethods.SetAALevel(this.NativeContext, value, -1, -1);
00099         }
00100     }
00101
00102 /// <summary>
00103 /// Gets or sets the current text anti-aliasing level.
00104 /// </summary>
00105     public int TextAntiAliasing
00106     {
00107         get
00108         {
00109             NativeMethods.GetAALevel(this.NativeContext, out _, out _, out int tbr);
00110             return tbr;
00111         }
00112
00113         set
00114         {
00115             if (value < 0 || value > 8)
00116             {
00117                 throw new ArgumentOutOfRangeException(nameof(value), value, "The anti-aliasing
level must range between 0 and 8 (inclusive).");
00118             }
00119
00120             NativeMethods.SetAALevel(this.NativeContext, -1, -1, value);
00121         }
00122     }
00123
00124 /// <summary>
00125 /// Gets or sets the current graphics anti-aliasing level.
00126 /// </summary>
00127     public int GraphicsAntiAliasing
00128     {
00129         get
00130         {
00131             NativeMethods.GetAALevel(this.NativeContext, out _, out int tbr, out _);
00132             return tbr;
00133         }
00134
00135         set
00136         {
00137             if (value < 0 || value > 8)
00138             {
00139                 throw new ArgumentOutOfRangeException(nameof(value), value, "The anti-aliasing
level must range between 0 and 8 (inclusive).");
00140             }
00141
00142             int prevTxt = this.TextAntiAliasing;
```

```

00143             NativeMethods.SetAALevel(this.NativeContext, -1, value, prevTxt);
00144         }
00145     }
00146
00147 /// <summary>
00148 /// Create a new <see cref="MuPDFContext"/> instance with the specified cache store size.
00149 /// </summary>
00150 /// <param name="storeSize">The maximum size in bytes of the resource cache store. The default value
00151     is 256 MiB.</param>
00152     public MuPDFContext(uint storeSize = 256 * 20)
00153     {
00154         ExitCodes result = (ExitCodes)NativeMethods.CreateContext((ulong)storeSize, ref
00155             NativeContext);
00156         switch (result)
00157         {
00158             case ExitCodes.EXIT_SUCCESS:
00159                 break;
00160             case ExitCodes.ERR_CANNOT_CREATE_CONTEXT:
00161                 throw new MuPDFException("Cannot create MuPDF context", result);
00162             case ExitCodes.ERR_CANNOT_REGISTER_HANDLERS:
00163                 throw new MuPDFException("Cannot register document handlers", result);
00164             default:
00165                 throw new MuPDFException("Unknown error", result);
00166         }
00167
00168 /// <summary>
00169 /// Wrap an existing pointer to a native MuPDF context object.
00170 /// </summary>
00171 /// <param name="parentContext">The parent context for cloned contexts.</param>
00172 /// <param name="nativeContext">The pointer to the native context that should be used.</param>
00173     internal MuPDFContext(MuPDFContext parentContext, IntPtr nativeContext)
00174     {
00175         this.NativeContext = nativeContext;
00176         this.ParentContext = parentContext;
00177     }
00178
00179 /// <summary>
00180 /// Evict all items from the resource cache store (freeing the memory where they were held).
00181 /// </summary>
00182     public void ClearStore()
00183     {
00184         NativeMethods.EmptyStore(this.NativeContext);
00185     }
00186
00187 /// <summary>
00188 /// Evict items from the resource cache store (freeing the memory where they were held) until the the
00189     size of the store drops to the specified fraction of the current size.
00190 /// </summary>
00191 /// <param name="fraction">The fraction of the current size that constitutes the target size of the
00192     store. If this is &lt;= 0, the cache is cleared. If this is &gt;= 1, nothing happens.</param>
00193     public void ShrinkStore(double fraction)
00194     {
00195         if (fraction <= 0)
00196         {
00197             ClearStore();
00198         }
00199         else if (Math.Round(fraction * 100) < 100)
00200         {
00201             NativeMethods.ShrinkStore(this.NativeContext, (uint)Math.Round(fraction * 100));
00202         }
00203
00204 /// <summary>
00205 /// Resolve a font from the font cache, or create a new font object.
00206 /// </summary>
00207 /// <param name="nativePointer">The pointer to the font.</param>
00208 /// <returns>The cached font instance, or a new font object.</returns>
00209     internal MuPDFFont Resolve(IntPtr nativePointer)
00210     {
00211         lock (FontCacheLock)
00212         {
00213             (MuPDFFont font, int referenceCount) item;
00214
00215             if (!this.FontCache.TryGetValue(nativePointer, out item) || item.font.disposedValue)
00216             {
00217                 item = (new MuPDFFont(this, nativePointer), 1);
00218                 this.FontCache[nativePointer] = item;
00219             }
00220             else
00221             {
00222                 item = (item.font, item.referenceCount + 1);
00223                 this.FontCache[nativePointer] = item;
00224             }
00225
00226             return item.font;
00227         }
00228     }

```

```

00226             }
00227         }
00228
00229         internal bool disposedValue;
00230
00231 ///<inheritdoc>
00232         protected virtual void Dispose(bool disposing)
00233     {
00234         if (!disposedValue)
00235         {
00236             if (disposing && this.ParentContext == null)
00237             {
00238                 lock (fontCacheLock)
00239                 {
00240                     this.fontCache.Clear();
00241                 }
00242
00243                 this.fontCache = null;
00244             }
00245
00246             NativeMethods.DisposeContext(NativeContext);
00247             disposedValue = true;
00248         }
00249     }
00250
00251 ///<inheritdoc>
00252         ~MuPDFContext()
00253     {
00254         Dispose(disposing: false);
00255     }
00256
00257 ///<inheritdoc>
00258         public void Dispose()
00259     {
00260         Dispose(disposing: true);
00261         GC.SuppressFinalize(this);
00262     }
00263 }
00264 }
```

8.7 MuPDFDisplayList.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019
00020 namespace MuPDFCore
00021 {
00022 /// <summary>
00023 /// A wrapper around a MuPDF display list object, which contains the necessary informations to render
00024 /// a page to an image.
00025 /// </summary>
00026         internal class MuPDFDisplayList : IDisposable
00027 /// <summary>
00028 /// The context that owns the document that was used to create this display list.
00029 /// </summary>
00030         private readonly MuPDFContext OwnerContext;
00031
00032 /// <summary>
00033 /// A pointer to the native display list object.
00034 /// </summary>
00035         readonly internal IntPtr NativeDisplayList;
00036
00037 /// <summary>
00038 /// The display list's bounds in page units. Read-only.
00039 /// </summary>
00040         public Rectangle Bounds { get; }
```

```

00041
00042 /// <summary>
00043 /// Create a new <see cref="MuPDFDisplayList"/> instance from the specified page.
00044 /// </summary>
00045 /// <param name="context">The context that owns the document from which the page was taken.</param>
00046 /// <param name="page">The page from which the display list should be generated.</param>
00047 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00048     signatures) are included in the display list that is generated. Otherwise, only the page contents are
00049     included.</param>
00050     public MuPDFDisplayList(MuPDFContext context, MuPDFPage page, bool includeAnnotations = true)
00051     {
00052         this.OwnerContext = context;
00053         float x0 = 0;
00054         float y0 = 0;
00055         float x1 = 0;
00056         float y1 = 0;
00057         ExitCodes result = (ExitCodes)NativeMethods.GetDisplayList(context.NativeContext,
00058             page.NativePage, includeAnnotations ? 1 : 0, ref NativeDisplayList, ref x0, ref y0, ref x1, ref y1);
00059         switch (result)
00060         {
00061             case ExitCodes.EXIT_SUCCESS:
00062                 break;
00063             case ExitCodes.ERR_CANNOT_RENDER:
00064                 throw new MuPDFException("Cannot render page", result);
00065             default:
00066                 throw new MuPDFException("Unknown error", result);
00067         }
00068         this.Bounds = new Rectangle(Math.Round(x0 * page.OwnerDocument.ImageXRes / 72.0 * 1000) /
00069             1000, Math.Round(y0 * page.OwnerDocument.ImageYRes / 72.0 * 1000) / 1000, Math.Round(x1 *
00070             page.OwnerDocument.ImageXRes / 72.0 * 1000) / 1000, Math.Round(y1 * page.OwnerDocument.ImageYRes /
00071             72.0 * 1000) / 1000);
00072         }
00073     private bool disposedValue;
00074     protected virtual void Dispose(bool disposing)
00075     {
00076         if (!disposedValue)
00077         {
00078             NativeMethods.DisposeDisplayList(OwnerContext.NativeContext, NativeDisplayList);
00079             disposedValue = true;
00080         }
00081     }
00082     ~MuPDFDisplayList()
00083     {
00084         Dispose(disposing: false);
00085     }
00086     public void Dispose()
00087     {
00088         Dispose(disposing: true);
00089         GC.SuppressFinalize(this);
00090     }
00091 }
00092 }
```

8.8 MuPDFDocument.Create.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.IO;
00004 using System.Linq;
00005 using System.Runtime.InteropServices.ComTypes;
00006 using System.Text;
00007
00008 namespace MuPDFCore
00009 {
00010 /// <summary>
00011 /// Options for creating a PDF document.
00012 /// </summary>
00013     public class PDFCreationOptions
00014     {
00015     /// <summary>
00016     /// Stream compression options for PDF documents.
00017     /// </summary>
00018         public enum CompressionOptions
00019         {
00020     /// <summary>
```

```
00021 /// Use the default settings for each stream.
00022 /// </summary>
00023     Preserve = 0,
00024
00025 /// <summary>
00026 /// Decompress all streams (except if <see cref="CompressFonts"/> or <see cref="CompressImages"/> are
00027 /// also specified; cannot be specified at the same time as <see cref="Compress"/>).
00028     Decompress = 0b1,
00029
00030 /// <summary>
00031 /// Compress all streams (cannot be specified at the same time as <see cref="Decompress"/>).
00032 /// </summary>
00033     Compress = 0b10,
00034
00035 /// <summary>
00036 /// Compress embedded fonts.
00037 /// </summary>
00038     CompressFonts = 0b100,
00039
00040 /// <summary>
00041 /// Compress images.
00042 /// </summary>
00043     CompressImages = 0b1000
00044 }
00045
00046 /// <summary>
00047 /// Options for garbage collection.
00048 /// </summary>
00049     public enum GarbageCollectionOption
00050     {
00051 /// <summary>
00052 /// Do not perform any garbage collection.
00053 /// </summary>
00054     None,
00055
00056 /// <summary>
00057 /// Garbage collect unused objects.
00058 /// </summary>
00059     Collect,
00060
00061 /// <summary>
00062 /// Garbage collect unused objects and compact XRef table.
00063 /// </summary>
00064     CollectCompact,
00065
00066 /// <summary>
00067 /// Garbage collect unused objects, compact XRef table, and remove duplicate objects.
00068 /// </summary>
00069     CollectCompactDeduplicate
00070 }
00071
00072 /// <summary>
00073 /// Document encryption options.
00074 /// </summary>
00075     public enum Encryption
00076     {
00077 /// <summary>
00078 /// Create an unencrypted document.
00079 /// </summary>
00080     None,
00081
00082 /// <summary>
00083 /// RC4 cipher with 40-bit key (a <see cref="UserPassword"/> or <see cref="OwnerPassword"/> must be
00084 /// supplied).
00085     RC4_40,
00086
00087 /// <summary>
00088 /// RC4 cipher with 128-bit key (a <see cref="UserPassword"/> or <see cref="OwnerPassword"/> must be
00089 /// supplied).
00090     RC4_128,
00091
00092 /// <summary>
00093 /// AES cipher with 128-bit key (a <see cref="UserPassword"/> or <see cref="OwnerPassword"/> must be
00094 /// supplied).
00095     AES_128,
00096
00097 /// <summary>
00098 /// AES cipher with 256-bit key (a <see cref="UserPassword"/> or <see cref="OwnerPassword"/> must be
00099 /// supplied).
00100     AES_256
00101 }
```

```
00103 /// <summary>
00104 /// Specifies which actions can be performed without the owner password.
00105 /// </summary>
00106     public class DocumentPermissions
00107     {
00108     /// <summary>
00109     /// Quality of allowed printing operations.
00110     /// </summary>
00111         public enum PrintingPermission
00112         {
00113     /// <summary>
00114     /// The document cannot be printed without the owner password.
00115     /// </summary>
00116         None,
00117
00118     /// <summary>
00119     /// Allow printing, but only in a low-level representation, possibly of degraded quality.
00120     /// </summary>
00121         LowFidelity,
00122
00123     /// <summary>
00124     /// Allow printing to a representation from which a faithful digital copy of the document can be
00125     /// generated.
00126         HighFidelity
00127     }
00128
00129     /// <summary>
00130     /// Allowed text and graphics copy/extraction operations.
00131     /// </summary>
00132         public enum ExtractionPermission
00133         {
00134     /// <summary>
00135     /// Text and graphics cannot be extracted or copied from the document without the owner password.
00136     /// </summary>
00137         None,
00138
00139     /// <summary>
00140     /// Text and graphics can be extracted from the document (for accessibility or other purposes).
00141     /// </summary>
00142         ExtractTextAndGraphics,
00143
00144     /// <summary>
00145     /// Text and graphics can be extracted and copied from the document.
00146     /// </summary>
00147         CopyAndExtractTextAndGraphics
00148     }
00149
00150     /// <summary>
00151     /// Allowed form and annotation operations.
00152     /// </summary>
00153         public enum FormPermission
00154         {
00155     /// <summary>
00156     /// Forms cannot be filled in without the owner password, and annotations cannot be added.
00157     /// </summary>
00158         None,
00159
00160     /// <summary>
00161     /// Forms can be filled in without the owner password, but annotations cannot be added.
00162     /// </summary>
00163         Forms,
00164
00165     /// <summary>
00166     /// Forms can be filled in without the owner password, and annotations can be added.
00167     /// </summary>
00168         FormsAnnotations
00169     }
00170
00171     /// <summary>
00172     /// Specifies if printing is allowed without the owner password.
00173     /// </summary>
00174         public PrintingPermission Printing { get; set; } = PrintingPermission.HighFidelity;
00175
00176     /// <summary>
00177     /// Specifies whether text can be extracted without the owner password.
00178     /// </summary>
00179         public ExtractionPermission Extraction { get; set; } =
00180             ExtractionPermission.CopyAndExtractTextAndGraphics;
00181
00182     /// <summary>
00183     /// Specifies whether the document can be restructured without the owner password (e.g., by inserting,
00184     /// rotating or deleting pages, bookmarks and thumbnail images).
00185     /// </summary>
00186         public bool AllowDocumentRestructuring { get; set; }
```

```
00187 // Specifies whether forms can be filled-in and annotations can be added to the document without the
00188 // owner password.
00189     public FormPermission Annotations { get; set; }
00190
00191 /// <summary>
00192 /// Specifies whether the document can be modified without the owner password.
00193 /// </summary>
00194     public bool AllowDocumentModification { get; set; }
00195
00196     internal int GetPermissionNumber()
00197     {
00198         uint tbr = 0b11111111111111111111000011000000;
00199
00200         switch (Printing)
00201         {
00202             case PrintingPermission.None:
00203                 break;
00204             case PrintingPermission.LowFidelity:
00205                 tbr |= 0b000000000100;
00206                 break;
00207             case PrintingPermission.HighFidelity:
00208                 tbr |= 0b100000000100;
00209                 break;
00210         }
00211
00212         switch (Extraction)
00213         {
00214             case ExtractionPermission.None:
00215                 break;
00216             case ExtractionPermission.ExtractTextAndGraphics:
00217                 tbr |= 0b001000000000;
00218                 break;
00219             case ExtractionPermission.CopyAndExtractTextAndGraphics:
00220                 tbr |= 0b001000010000;
00221                 break;
00222         }
00223
00224         if (AllowDocumentRestructuring)
00225         {
00226             tbr |= 0b010000000000;
00227         }
00228
00229         switch (Annotations)
00230         {
00231             case FormPermission.None:
00232                 break;
00233             case FormPermission.Forms:
00234                 tbr |= 0b000100000000;
00235                 break;
00236             case FormPermission.FormsAnnotations:
00237                 tbr |= 0b000100100000;
00238                 break;
00239         }
00240
00241         if (AllowDocumentModification)
00242         {
00243             tbr |= 0b000000001000;
00244         }
00245
00246         return (int)tbr;
00247     }
00248 }
00249
00250 /// <summary>
00251 /// If this is <see langword="true" />, annotations (e.g. signatures) are included in the converted
00252 // document. Otherwise, only the page contents are included.
00253 /// </summary>
00254     public bool IncludeAnnotations { get; set; } = true;
00255
00256     private CompressionOptions compressStreams = CompressionOptions.Preserve;
00257
00258 /// <summary>
00259 /// Determines whether streams are compressed in the generated PDF document.
00260 /// </summary>
00261     public CompressionOptions CompressStreams
00262     {
00263         get => compressStreams;
00264         set
00265         {
00266             if (value.HasFlag(CompressionOptions.Compress) &&
00267                 value.HasFlag(CompressionOptions.Decompress))
00268             {
00269                 throw new ArgumentException("CompressionOptions.Compress and
CompressionOptions.Decompress may not be specified at the same time!", "value");
00270             }
00271         }
00272     }
```

```
00270             this.compressStreams = value;
00271         }
00272     }
00273
00274 /// <summary>
00275 /// Use ASCII hex encoding for binary streams.
00276 /// </summary>
00277     public bool AsciiEncode { get; set; } = false;
00278
00279 /// <summary>
00280 /// Pretty-print objects with indentation.
00281 /// </summary>
00282     public bool PrettyPrint { get; set; } = false;
00283
00284 /// <summary>
00285 /// Generate a linearized PDF, optimized for loading in web browsers.
00286 /// </summary>
00287     public bool Linearize { get; set; } = false;
00288
00289 /// <summary>
00290 /// Pretty-print graphics commands in content streams.
00291 /// </summary>
00292     public bool Clean { get; set; } = false;
00293
00294 /// <summary>
00295 /// Sanitize graphics commands in content streams.
00296 /// </summary>
00297     public bool Sanitize { get; set; } = false;
00298
00299 /// <summary>
00300 /// Garbage collection options.
00301 /// </summary>
00302     public GarbageCollectionOption Garbage { get; set; } = GarbageCollectionOption.None;
00303
00304 /// <summary>
00305 /// Type of encryption to use.
00306 /// </summary>
00307     public Encryption EncryptionType { get; set; } = Encryption.None;
00308
00309 /// <summary>
00310 /// User password required to read the newly created document.
00311 /// </summary>
00312     public string UserPassword { get; set; } = null;
00313
00314 /// <summary>
00315 /// Owner password required to edit the newly created document.
00316 /// </summary>
00317     public string OwnerPassword { get; set; } = null;
00318
00319 /// <summary>
00320 /// Actions that can be performed without the owner password (an <see cref="OwnerPassword"/> must be supplied, and an <see cref="EncryptionType"/> must be specified.
00321 /// </summary>
00322     public DocumentPermissions Permissions { get; set; } = null;
00323
00324 /// <summary>
00325 /// Regenerate document id.
00326 /// </summary>
00327     public bool RegenerateId { get; set; } = true;
00328
00329     internal string GetOptionString()
00330     {
00331         StringBuilder sb = new StringBuilder();
00332
00333         if (this.CompressStreams.HasFlag(CompressionOptions.Compress))
00334         {
00335             if (sb.Length > 0)
00336             {
00337                 sb.Append(",");
00338             }
00339             sb.Append("compress=yes");
00340         }
00341         else if (this.CompressStreams.HasFlag(CompressionOptions.Decompress))
00342         {
00343             if (sb.Length > 0)
00344             {
00345                 sb.Append(",");
00346             }
00347             sb.Append("decompress=yes");
00348         }
00349
00350         if (this.CompressStreams.HasFlag(CompressionOptions.CompressFonts))
00351         {
00352             if (sb.Length > 0)
00353             {
00354                 sb.Append(",");
00355             }
00356         }
00357     }
00358 }
```

```
00356             sb.Append("compress-fonts=yes");
00357         }
00358
00359         if (this.CompressStreams.HasFlag(CompressionOptions.CompressImages))
00360     {
00361         if (sb.Length > 0)
00362         {
00363             sb.Append(",");
00364         }
00365         sb.Append("compress-images=yes");
00366     }
00367
00368         if (this.AsciiEncode)
00369     {
00370         if (sb.Length > 0)
00371         {
00372             sb.Append(",");
00373         }
00374         sb.Append("ascii=yes");
00375     }
00376
00377         if (this.PrettyPrint)
00378     {
00379         if (sb.Length > 0)
00380         {
00381             sb.Append(",");
00382         }
00383         sb.Append("pretty=yes");
00384     }
00385
00386         if (this.Linearize)
00387     {
00388         if (sb.Length > 0)
00389         {
00390             sb.Append(",");
00391         }
00392         sb.Append("linearize=yes");
00393     }
00394
00395         if (this.Clean)
00396     {
00397         if (sb.Length > 0)
00398         {
00399             sb.Append(",");
00400         }
00401         sb.Append("clean=yes");
00402     }
00403
00404         if (this.Sanitize)
00405     {
00406         if (sb.Length > 0)
00407         {
00408             sb.Append(",");
00409         }
00410         sb.Append("sanitize=yes");
00411     }
00412
00413         switch (this.Garbage)
00414     {
00415             case GarbageCollectionOption.None:
00416                 break;
00417             case GarbageCollectionOption.Collect:
00418                 if (sb.Length > 0)
00419                 {
00420                     sb.Append(",");
00421                 }
00422                 sb.Append("garbage=yes");
00423                 break;
00424             case GarbageCollectionOption.CollectCompact:
00425                 if (sb.Length > 0)
00426                 {
00427                     sb.Append(",");
00428                 }
00429                 sb.Append("garbage=compact");
00430                 break;
00431             case GarbageCollectionOption.CollectCompactDeduplicate:
00432                 if (sb.Length > 0)
00433                 {
00434                     sb.Append(",");
00435                 }
00436                 sb.Append("garbage=deduplicate");
00437                 break;
00438             }
00439
00440             if (this.EncryptionType == Encryption.None)
00441             {
00442                 if (sb.Length > 0)
```

```
00443             {
00444                 sb.Append(",");
00445             }
00446             sb.Append("decrypt=yes");
00447
00448             if (Permissions != null)
00449             {
00450                 throw new InvalidOperationException("Encryption must be enabled and an owner
password must be supplied in order to use permissions.");
00451             }
00452
00453             if (!string.IsNullOrEmpty(OwnerPassword))
00454             {
00455                 throw new InvalidOperationException("Encryption must be enabled and a set of
permissions must be supplied in order to use an owner password.");
00456             }
00457
00458             if (!string.IsNullOrEmpty(UserPassword))
00459             {
00460                 throw new InvalidOperationException("Encryption must be enabled in order to use a
user password.");
00461             }
00462         }
00463         else
00464         {
00465             if (sb.Length > 0)
00466             {
00467                 sb.Append(",");
00468             }
00469
00470             switch (this.EncryptionType)
00471             {
00472                 case Encryption.RC4_40:
00473                     sb.Append("encrypt=rc4-40");
00474                     break;
00475
00476                 case Encryption.RC4_128:
00477                     sb.Append("encrypt=rc4-128");
00478                     break;
00479
00480                 case Encryption.AES_128:
00481                     sb.Append("encrypt=aes-128");
00482                     break;
00483
00484                 case Encryption.AES_256:
00485                     sb.Append("encrypt=aes-256");
00486                     break;
00487             }
00488
00489             if (string.IsNullOrEmpty(UserPassword) && string.IsNullOrEmpty(OwnerPassword))
00490             {
00491                 throw new InvalidOperationException("A user password or an owner password must be
supplied in order to use encryption.");
00492             }
00493         }
00494
00495         if (!string.IsNullOrEmpty(UserPassword))
00496         {
00497             if (sb.Length > 0)
00498             {
00499                 sb.Append(",");
00500             }
00501             sb.Append("user-password=");
00502             sb.Append(UserPassword);
00503         }
00504
00505         if (!string.IsNullOrEmpty(OwnerPassword))
00506         {
00507             if (sb.Length > 0)
00508             {
00509                 sb.Append(",");
00510             }
00511             sb.Append("owner-password=");
00512             sb.Append(OwnerPassword);
00513         }
00514
00515         if (Permissions != null)
00516         {
00517             if (string.IsNullOrEmpty(OwnerPassword))
00518             {
00519                 throw new InvalidOperationException("Encryption must be enabled and an owner
password must be supplied in order to use permissions.");
00520             }
00521
00522             if (sb.Length > 0)
00523             {
00524                 sb.Append(",");
00525             }
00526         }
00527     }
00528 }
```

```
00525             }
00526             sb.Append("permissions=");
00527         sb.Append(this.Permissions.GetPermissionNumber().ToString(System.Globalization.CultureInfo.InvariantCulture));
00528     }
00529
00530     if (!this.RegenerateId)
00531     {
00532         if (sb.Length > 0)
00533         {
00534             sb.Append(",");
00535         }
00536         sb.Append("regenerate-id=no");
00537     }
00538
00539     return sb.ToString();
00540 }
00541 }
00542
00543 /// <summary>
00544 /// Options for creating an SVG document.
00545 /// </summary>
00546 public class SVGCreationOptions
00547 {
00548 /// <summary>
00549 /// Ways in which text is represented in the SVG document.
00550 /// </summary>
00551     public enum TextOption
00552     {
00553     /// <summary>
00554     /// Emit text as &lt;text&ampgt elements (inaccurate fonts).
00555     /// </summary>
00556         TextAsText,
00557
00558     /// <summary>
00559     /// Emit text as &lt;path&ampgt elements (accurate fonts, but text is not searchable or selectable).
00560     /// </summary>
00561         TextAsPath
00562     }
00563
00564     /// <summary>
00565     /// If this is <see langword="true" />, annotations (e.g. signatures) are included in the converted
00566     /// document. Otherwise, only the page contents are included.
00567     public bool IncludeAnnotations { get; set; } = true;
00568
00569     /// <summary>
00570     /// Determines how text is represented in the SVG document.
00571     /// </summary>
00572     public TextOption TextRendering { get; set; } = TextOption.TextAsPath;
00573
00574     /// <summary>
00575     /// Whether to reuse images using &lt;symbol&ampgt definitions.
00576     /// </summary>
00577     public bool ReuseImages { get; set; } = true;
00578
00579     internal string GetOptionString()
00580     {
00581         StringBuilder sb = new StringBuilder();
00582
00583         if (sb.Length > 0)
00584         {
00585             sb.Append(",");
00586         }
00587         switch (this.TextRendering)
00588         {
00589             case TextOption.TextAsPath:
00590                 sb.Append("text=path");
00591                 break;
00592             case TextOption.TextAsText:
00593                 sb.Append("text=text");
00594                 break;
00595         }
00596
00597         if (!ReuseImages)
00598         {
00599             if (sb.Length > 0)
00600             {
00601                 sb.Append(",");
00602             }
00603             sb.Append("no-reuse-images=yes");
00604         }
00605
00606         return sb.ToString();
00607     }
00608 }
```

```
00610 /// <summary>
00611 /// Options for creating a CBZ document.
00612 /// </summary>
00613     public class CBZCreationOptions
00614     {
00615     /// <summary>
00616     /// If this is <see langword="true" />, annotations (e.g. signatures) are included in the converted
00617     /// document. Otherwise, only the page contents are included.
00618         public bool IncludeAnnotations { get; set; } = true;
00619
00620     /// <summary>
00621     /// Colour spaces for the rendered images.
00622     /// </summary>
00623         public enum ColorSpace
00624         {
00625     /// <summary>
00626     /// Grayscale
00627     /// </summary>
00628         Gray,
00629
00630     /// <summary>
00631     /// RGB colour space.
00632     /// </summary>
00633         RGB,
00634
00635     /// <summary>
00636     /// CMYK colour space.
00637     /// </summary>
00638         CMYK
00639     }
00640
00641     /// <summary>
00642     /// Rasterizer settings.
00643     /// </summary>
00644         public enum RasterizerOption
00645         {
00646     /// <summary>
00647     /// Default rasterizer.
00648     /// </summary>
00649         Default,
00650
00651     /// <summary>
00652     /// Anti-aliasing with 0 bits.
00653     /// </summary>
00654         AntiAliasing_0,
00655
00656     /// <summary>
00657     /// Anti-aliasing with 1 bit.
00658     /// </summary>
00659         AntiAliasing_1,
00660
00661     /// <summary>
00662     /// Anti-aliasing with 2 bits.
00663     /// </summary>
00664         AntiAliasing_2,
00665
00666     /// <summary>
00667     /// Anti-aliasing with 3 bits.
00668     /// </summary>
00669         AntiAliasing_3,
00670
00671     /// <summary>
00672     /// Anti-aliasing with 4 bits.
00673     /// </summary>
00674         AntiAliasing_4,
00675
00676     /// <summary>
00677     /// Anti-aliasing with 5 bits.
00678     /// </summary>
00679         AntiAliasing_5,
00680
00681     /// <summary>
00682     /// Anti-aliasing with 6 bits.
00683     /// </summary>
00684         AntiAliasing_6,
00685
00686     /// <summary>
00687     /// Anti-aliasing with 7 bits.
00688     /// </summary>
00689         AntiAliasing_7,
00690
00691     /// <summary>
00692     /// Anti-aliasing with 8 bits.
00693     /// </summary>
00694         AntiAliasing_8,
00695
```

```
00696 /// <summary>
00697 /// Centre of pixel.
00698 /// </summary>
00699     CenterOfPixel,
00700
00701 /// <summary>
00702 /// Any part of the pixel.
00703 /// </summary>
00704     AnyPartOfPixel
00705 }
00706
00707
00708 /// <summary>
00709 /// Angle (in degrees) by which the rendered pages will be rotated.
00710 /// </summary>
00711     public double Rotate { get; set; } = double.NaN;
00712
00713 /// <summary>
00714 /// X resolution of the rendered pages in pixels per inch.
00715 /// </summary>
00716     public double XResolution { get; set; } = double.NaN;
00717
00718 /// <summary>
00719 /// Y resolution of the rendered pages in pixels per inch.
00720 /// </summary>
00721     public double YResolution { get; set; } = double.NaN;
00722
00723 /// <summary>
00724 /// Render pages to fit the specified width.
00725 /// </summary>
00726     public double Width { get; set; } = double.NaN;
00727
00728 /// <summary>
00729 /// Render pages to fit the specified height.
00730 /// </summary>
00731     public double Height { get; set; } = double.NaN;
00732
00733 /// <summary>
00734 /// Colour space for rendering.
00735 /// </summary>
00736     public ColorSpace RenderingColorSpace { get; set; } = ColorSpace.RGB;
00737
00738 /// <summary>
00739 /// Render pages with an alpha channel and transparent background.
00740 /// </summary>
00741     public bool Alpha { get; set; } = false;
00742
00743 /// <summary>
00744 /// Rasterizer settings for graphics elements.
00745 /// </summary>
00746     public RasterizerOption GraphicsRasterizer { get; set; } = RasterizerOption.Default;
00747
00748 /// <summary>
00749 /// Rasterizer settings for text elements.
00750 /// </summary>
00751     public RasterizerOption TextRasterizer { get; set; } = RasterizerOption.Default;
00752
00753     internal string GetOptionString()
00754     {
00755         StringBuilder sb = new StringBuilder();
00756
00757         if (!double.IsNaN(Rotate))
00758         {
00759             if (sb.Length > 0)
00760             {
00761                 sb.Append(",");
00762             }
00763             sb.Append("rotate=" +
00764             Rotate.ToString(System.Globalization.CultureInfo.InvariantCulture));
00765         }
00766
00767         if (!double.IsNaN(XResolution))
00768         {
00769             if (sb.Length > 0)
00770             {
00771                 sb.Append(",");
00772             }
00773             sb.Append("x-resolution=" +
00774             XResolution.ToString(System.Globalization.CultureInfo.InvariantCulture));
00775         }
00776
00777         if (!double.IsNaN(YResolution))
00778         {
00779             if (sb.Length > 0)
00780             {
```

```
00781             sb.Append("y-resolution=" +
00782             YResolution.ToString(System.Globalization.CultureInfo.InvariantCulture));
00783         }
00784         if (!double.IsNaN(Width))
00785         {
00786             if (sb.Length > 0)
00787             {
00788                 sb.Append(",");
00789             }
00790             sb.Append("width=" +
00791             Width.ToString(System.Globalization.CultureInfo.InvariantCulture));
00792         }
00793         if (!double.IsNaN(Height))
00794         {
00795             if (sb.Length > 0)
00796             {
00797                 sb.Append(",");
00798             }
00799             sb.Append("height=" +
00800             Height.ToString(System.Globalization.CultureInfo.InvariantCulture));
00801         }
00802         if (sb.Length > 0)
00803         {
00804             sb.Append(",");
00805         }
00806
00807         switch (this.RenderingColorSpace)
00808     {
00809         case ColorSpace.Gray:
00810             sb.Append("colorspace=gray");
00811             break;
00812         case ColorSpace.RGB:
00813             sb.Append("colorspace=rgb");
00814             break;
00815         case ColorSpace.CMYK:
00816             sb.Append("colorspace=cmyk");
00817             break;
00818     }
00819
00820     if (Alpha)
00821     {
00822         if (sb.Length > 0)
00823         {
00824             sb.Append(",");
00825         }
00826         sb.Append("alpha=yes");
00827     }
00828     else
00829     {
00830         if (sb.Length > 0)
00831         {
00832             sb.Append(",");
00833         }
00834         sb.Append("alpha=no");
00835     }
00836
00837     if (GraphicsRasterizer != RasterizerOption.Default)
00838     {
00839         if (sb.Length > 0)
00840         {
00841             sb.Append(",");
00842         }
00843
00844         switch (GraphicsRasterizer)
00845     {
00846         case RasterizerOption.AntiAliasing_0:
00847             sb.Append("graphics=aa0");
00848             break;
00849         case RasterizerOption.AntiAliasing_1:
00850             sb.Append("graphics=aa1");
00851             break;
00852         case RasterizerOption.AntiAliasing_2:
00853             sb.Append("graphics=aa2");
00854             break;
00855         case RasterizerOption.AntiAliasing_3:
00856             sb.Append("graphics=aa3");
00857             break;
00858         case RasterizerOption.AntiAliasing_4:
00859             sb.Append("graphics=aa4");
00860             break;
00861         case RasterizerOption.AntiAliasing_5:
00862             sb.Append("graphics=aa5");
00863             break;
00864         case RasterizerOption.AntiAliasing_6:
```

```

00865             sb.Append("graphics=aa6");
00866             break;
00867         case RasterizerOption.AntiAliasing_7:
00868             sb.Append("graphics=aa7");
00869             break;
00870         case RasterizerOption.AntiAliasing_8:
00871             sb.Append("graphics=aa8");
00872             break;
00873         case RasterizerOption.CenterOfPixel:
00874             sb.Append("graphics=cop");
00875             break;
00876         case RasterizerOption.AnyPartOfPixel:
00877             sb.Append("graphics=app");
00878             break;
00879         }
00880     }
00881
00882     if (TextRasterizer != RasterizerOption.Default)
00883     {
00884         if (sb.Length > 0)
00885         {
00886             sb.Append(",");
00887         }
00888
00889         switch (TextRasterizer)
00890         {
00891             case RasterizerOption.AntiAliasing_0:
00892                 sb.Append("text=aa0");
00893                 break;
00894             case RasterizerOption.AntiAliasing_1:
00895                 sb.Append("text=aal");
00896                 break;
00897             case RasterizerOption.AntiAliasing_2:
00898                 sb.Append("text=aa2");
00899                 break;
00900             case RasterizerOption.AntiAliasing_3:
00901                 sb.Append("text=aa3");
00902                 break;
00903             case RasterizerOption.AntiAliasing_4:
00904                 sb.Append("text=aa4");
00905                 break;
00906             case RasterizerOption.AntiAliasing_5:
00907                 sb.Append("text=aa5");
00908                 break;
00909             case RasterizerOption.AntiAliasing_6:
00910                 sb.Append("text=aa6");
00911                 break;
00912             case RasterizerOption.AntiAliasing_7:
00913                 sb.Append("text=aa7");
00914                 break;
00915             case RasterizerOption.AntiAliasing_8:
00916                 sb.Append("text=aa8");
00917                 break;
00918             case RasterizerOption.CenterOfPixel:
00919                 sb.Append("text=cop");
00920                 break;
00921             case RasterizerOption.AnyPartOfPixel:
00922                 sb.Append("text=app");
00923                 break;
00924         }
00925     }
00926
00927     return sb.ToString();
00928 }
00929 }
00930
00931 /// <summary>
00932 /// Options for creating a text or structured text document.
00933 /// </summary>
00934 public class TXTCreationOptions
00935 {
00936 /// <summary>
00937 /// If this is <see langword="true" />, annotations (e.g. signatures) are included in the converted
00938 /// document. Otherwise, only the page contents are included.
00939     public bool IncludeAnnotations { get; set; } = true;
00940
00941 /// <summary>
00942 /// Do not add spaces between gaps in the text.
00943 /// </summary>
00944     public bool InhibitSpaces { get; set; } = false;
00945
00946 /// <summary>
00947 /// Do not expand ligatures into constituent characters.
00948 /// </summary>
00949     public bool PreserveLigatures { get; set; } = false;
00950

```

```
00951 /// <summary>
00952 /// Do not convert all whitespace into space characters.
00953 /// </summary>
00954     public bool PreserveWhitespace { get; set; } = false;
00955
00956 /// <summary>
00957 /// Do not merge spans on the same line.
00958 /// </summary>
00959     public bool PreserveSpans { get; set; } = false;
00960
00961 /// <summary>
00962 /// Attempt to join up hyphenated words.
00963 /// </summary>
00964     public bool Dehyphenate { get; set; } = false;
00965
00966 /// <summary>
00967 /// Guess unicode from CID if normal mapping fails.
00968 /// </summary>
00969     public bool UseCIDForUnknownUnicode { get; set; } = false;
00970
00971 /// <summary>
00972 /// Include characters that are outside of the page's mediabox.
00973 /// </summary>
00974     public bool IncludeCharactersOutsideMediaBox { get; set; } = false;
00975
00976     internal string GetOptionString()
00977     {
00978         StringBuilder sb = new StringBuilder();
00979
00980         if (this.InhibitSpaces)
00981         {
00982             if (sb.Length > 0)
00983             {
00984                 sb.Append(",");
00985             }
00986             sb.Append("inhibit-spaces=yes");
00987         }
00988
00989         if (this.PreserveLigatures)
00990         {
00991             if (sb.Length > 0)
00992             {
00993                 sb.Append(",");
00994             }
00995             sb.Append("preserve-ligatures=yes");
00996         }
00997
00998         if (this.PreserveWhitespace)
00999         {
01000             if (sb.Length > 0)
01001             {
01002                 sb.Append(",");
01003             }
01004             sb.Append("preserve-whitespace=yes");
01005         }
01006
01007         if (this.PreserveSpans)
01008         {
01009             if (sb.Length > 0)
01010             {
01011                 sb.Append(",");
01012             }
01013             sb.Append("preserve-spans=yes");
01014         }
01015
01016         if (this.Dehyphenate)
01017         {
01018             if (sb.Length > 0)
01019             {
01020                 sb.Append(",");
01021             }
01022             sb.Append("dehyphenate=yes");
01023         }
01024
01025         if (this.UseCIDForUnknownUnicode)
01026         {
01027             if (sb.Length > 0)
01028             {
01029                 sb.Append(",");
01030             }
01031             sb.Append("use-cid-for-unknown-unicode=yes");
01032         }
01033
01034         if (this.IncludeCharactersOutsideMediaBox)
01035         {
01036             if (sb.Length > 0)
01037             {
```



```

01124         }
01125         sb.Append("preserve-ligatures=yes");
01126     }
01127
01128     if (this.PreserveWhitespace)
01129     {
01130         if (sb.Length > 0)
01131         {
01132             sb.Append(",");
01133         }
01134         sb.Append("preserve-whitespace=yes");
01135     }
01136
01137     if (this.PreserveSpans)
01138     {
01139         if (sb.Length > 0)
01140         {
01141             sb.Append(",");
01142         }
01143         sb.Append("preserve-spans=yes");
01144     }
01145
01146     if (this.Dehyphenate)
01147     {
01148         if (sb.Length > 0)
01149         {
01150             sb.Append(",");
01151         }
01152         sb.Append("dehyphenate=yes");
01153     }
01154
01155     if (this.UseCIDForUnknownUnicode)
01156     {
01157         if (sb.Length > 0)
01158         {
01159             sb.Append(",");
01160         }
01161         sb.Append("use-cid-for-unknown-unicode=yes");
01162     }
01163
01164     if (this.IncludeCharactersOutsideMediaBox)
01165     {
01166         if (sb.Length > 0)
01167         {
01168             sb.Append(",");
01169         }
01170         sb.Append("mediabox-clip=no");
01171     }
01172
01173     return sb.ToString();
01174 }
01175 }
01176
01177
01178 partial class MuPDFDocument
01179 {
01180 /// <summary>
01181 /// Create a new document containing the specified (parts of) pages from other documents.
01182 /// </summary>
01183 /// <param name="context">The context that was used to open the documents.</param>
01184 /// <param name="fileName">The output file name.</param>
01185 /// <param name="fileType">The output file format.</param>
01186 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01187 // signatures) are included in the display list that is generated. Otherwise, only the page contents are
01188 // included.</param>
01189 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01190 // the "region" element the area of the page that should be included in the document, and the "zoom"
01191 // element how much the region should be scaled.</param>
01192 [Obsolete("Please use MuPDFDocument.Create.Document instead, or one of the specific methods
01193 // for a document type in order to specify additional options.", false)]
01194 public static void CreateDocument(MuPDFContext context, string fileName,
01195 DocumentOutputFileTypes fileType, bool includeAnnotations = true, params (MuPDFPage page, Rectangle
01196 region, float zoom[])[] pages) => Create.Document(context, fileName, fileType, pages,
01197 includeAnnotations);
01198
01199 /// <summary>
01200 /// Create a new document containing the specified pages from other documents.
01201 /// </summary>
01202 /// <param name="context">The context that was used to open the documents.</param>
01203 /// <param name="fileName">The output file name.</param>
01204 /// <param name="fileType">The output file format.</param>
01205 /// <param name="pages">The pages to include in the document.</param>
01206 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01207 // signatures) are included in the display list that is generated. Otherwise, only the page contents are
01208 // included.</param>
01209 [Obsolete("Please use MuPDFDocument.Create.Document instead, or one of the specific methods
01210 // for a document type in order to specify additional options.", false)]

```

```

01200     public static void CreateDocument(MuPDFContext context, string fileName,
01201         DocumentOutputFileTypes fileType, bool includeAnnotations = true, params MuPDFPage[] pages) =>
01202     Create.Document(context, fileName, fileType, pages, includeAnnotations);
01203     /// <summary>
01204     /// Contains methods to create documents by combining pages from other documents.
01205     /// </summary>
01206     public static class Create
01207     {
01208         private static void CreateDocument(MuPDFContext context, string fileName,
01209             DocumentOutputFileTypes fileType, IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages,
01210             string optionString, bool includeAnnotations)
01211         {
01212             if (fileType == DocumentOutputFileTypes.SVG && pages.Count() > 1)
01213             {
01214                 //Actually, you can, but the library creates multiple files appending numbers
01215                 //after each name (e.g. page1.svg, page2.svg, ...), which is ugly and may have unintended consequences.
01216                 //If you really want to do this, you can call this method multiple times.
01217                 throw new ArgumentException("You cannot create an SVG document with more than one
01218                 page!", nameof(pages));
01219             }
01220         }
01221         string originalFileName = fileName;
01222         //For SVG documents, the library annoyingly alters the output file name, appending a
01223         // "1" just before the extension (e.g. document.svg -> document1.svg). Since users may not be expecting
01224         // this, it is best to render to a temporary file and then move it to the specified location.
01225         if (fileType == DocumentOutputFileTypes.SVG)
01226         {
01227             fileName = Path.GetTempFileName();
01228             File.Delete(fileName);
01229         }
01230         IntPtr documentWriter = IntPtr.Zero;
01231         ExitCodes result;
01232         // Encode the file name in UTF-8 in unmanaged memory.
01233         using (UTF8EncodedString encodedFileName = new UTF8EncodedString(fileName))
01234         using (UTF8EncodedString encodedOptions = new UTF8EncodedString(optionString))
01235         {
01236             //Initialise document writer.
01237             result = (ExitCodes)NativeMethods.CreateDocumentWriter(context.NativeContext,
01238             encodedFileName.Address, (int)fileType, encodedOptions.Address, ref documentWriter);
01239         }
01240         switch (result)
01241         {
01242             case ExitCodes.EXIT_SUCCESS:
01243                 break;
01244             case ExitCodes.ERR_CANNOT_CREATE_WRITER:
01245                 throw new MuPDFException("Cannot create the document writer", result);
01246             default:
01247                 throw new MuPDFException("Unknown error", result);
01248         }
01249         int i = 0;
01250         //Write pages.
01251         foreach ((MuPDFPage page, Rectangle region, float zoom) pag in pages)
01252         {
01253             MuPDFDocument doc = pag.page.OwnerDocument;
01254             int pageNum = pag.page.PageNumber;
01255             if (doc.DisplayLists[pageNum] == null)
01256             {
01257                 doc.DisplayLists[pageNum] = new MuPDFDisplayList(doc.OwnerContext,
01258                     doc.Pages[pageNum], includeAnnotations);
01259             }
01260             Rectangle region = pag.region;
01261             double zoom = pag.zoom;
01262             if (pag.page.OwnerDocument.ImageXRes != 72 || pag.page.OwnerDocument.ImageYRes !=
01263                 72)
01264             {
01265                 zoom *= Math.Sqrt(pag.page.OwnerDocument.ImageXRes *
01266                     pag.page.OwnerDocument.ImageYRes) / 72;
01267                 region = new Rectangle(region.X0 * 72 / pag.page.OwnerDocument.ImageXRes,
01268                     region.Y0 * 72 / pag.page.OwnerDocument.ImageYRes, region.X1 * 72 / pag.page.OwnerDocument.ImageXRes,
01269                     region.Y1 * 72 / pag.page.OwnerDocument.ImageYRes);
01270             }
01271             result = (ExitCodes)NativeMethods.WriteSubDisplayListAsPage(context.NativeContext,
01272                 doc.DisplayLists[pageNum].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, (float)zoom,
01273                 documentWriter);
01274     }

```

```

01271             switch (result)
01272             {
01273                 case ExitCodes.EXIT_SUCCESS:
01274                     break;
01275                 case ExitCodes.ERR_CANNOT_RENDER:
01276                     throw new MuPDFException("Cannot render page " + i.ToString(), result);
01277                 default:
01278                     throw new MuPDFException("Unknown error", result);
01279             }
01280             i++;
01281         }
01282     }
01283 
01284     //Close and dispose the document writer.
01285     result = (ExitCodes)NativeMethods.FinalizeDocumentWriter(context.NativeContext,
01286     documentWriter);
01287 
01288     switch (result)
01289     {
01290         case ExitCodes.EXIT_SUCCESS:
01291             break;
01292         case ExitCodes.ERR_CANNOT_CLOSE_DOCUMENT:
01293             throw new MuPDFException("Cannot finalise the document", result);
01294         default:
01295             throw new MuPDFException("Unknown error", result);
01296     }
01297 
01298     if (fileType == DocumentOutputFileTypes.SVG)
01299     {
01300         //Move the temporary file to the location specified by the user.
01301         //The library has altered the temporary file name by appending a "1" before the
01302         //extension.
01303         string tempFileName = Path.Combine(Path.GetDirectoryName(fileName),
01304         Path.GetFileNameWithoutExtension(fileName) + "1" + Path.GetExtension(fileName));
01305 
01306         //Overwrite existing file.
01307         if (File.Exists(originalFileName))
01308         {
01309             File.Delete(originalFileName);
01310         }
01311     }
01312 
01313 /// <summary>
01314 /// Create a new document containing the specified (parts of) pages from other documents.
01315 /// </summary>
01316 /// <param name="context">The context that was used to open the documents.</param>
01317 /// <param name="fileName">The output file name.</param>
01318 /// <param name="fileType">The output file format.</param>
01319 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01320         signatures) are included in the display list that is generated. Otherwise, only the page contents are
01321         included.</param>
01322 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01323         the "region" element the area of the page that should be included in the document, and the "zoom"
01324         element how much the region should be scaled.</param>
01325 
01326         public static void Document(MuPDFContext context, string fileName, DocumentOutputFileTypes
01327         fileType, IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, bool includeAnnotations =
01328             true) => CreateDocument(context, fileName, fileType, pages, "", includeAnnotations);
01329 
01330 /// <summary>
01331 /// Create a new document containing the specified (parts of) pages from other documents.
01332 /// </summary>
01333 /// <param name="context">The context that was used to open the documents.</param>
01334 /// <param name="fileName">The output file name.</param>
01335 /// <param name="fileType">The output file format.</param>
01336 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01337         the "region" element the area of the page that should be included in the document, and the "zoom"
01338         element how much the region should be scaled.</param>
01339 
01340         public static void Document(MuPDFContext context, string fileName, DocumentOutputFileTypes
01341         fileType, IEnumerable<MuPDFPage> pages, bool includeAnnotations = true) => Document(context, fileName,
01342         fileType, pages.Select(x => (x, x.Bounds, 1.0f)), includeAnnotations);

```

```

01341 /// <summary>
01342 /// Create a new document containing the specified (parts of) pages from other documents.
01343 /// </summary>
01344 /// <param name="context">The context that was used to open the documents.</param>
01345 /// <param name="fileName">The output file name.</param>
01346 /// <param name="fileType">The output file format.</param>
01347 /// <param name="pages">The pages to include in the document.</param>
01348 /// <param name="options">Options for the output format.</param>
01349     public static void Document(MuPDFContext context, string fileName, DocumentOutputFileTypes
      fileType, params MuPDFPage[] pages) => Document(context, fileName, fileType, pages, true);
01350
01351
01352 /// <summary>
01353 /// Create a new PDF document containing the specified (parts of) pages from other documents.
01354 /// </summary>
01355 /// <param name="context">The context that was used to open the documents.</param>
01356 /// <param name="fileName">The output file name.</param>
01357 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
  the "region" element the area of the page that should be included in the document, and the "zoom"
  element how much the region should be scaled.</param>
01358 /// <param name="options">Options for the output format.</param>
01359     public static void PDFDocument(MuPDFContext context, string fileName,
      IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, PDFCreationOptions options =
      default)
01360     {
01361         options = options ?? new PDFCreationOptions();
01362         string optionString = options.GetOptionString();
01363         CreateDocument(context, fileName, DocumentOutputFileTypes.PDF, pages, optionString,
          options.IncludeAnnotations);
01364     }
01365
01366 /// <summary>
01367 /// Create a new PDF document containing the specified (parts of) pages from other documents.
01368 /// </summary>
01369 /// <param name="context">The context that was used to open the documents.</param>
01370 /// <param name="fileName">The output file name.</param>
01371 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
  the "region" element the area of the page that should be included in the document, and the "zoom"
  element how much the region should be scaled.</param>
01372     public static void PDFDocument(MuPDFContext context, string fileName, params (MuPDFPage
      page, Rectangle region, float zoom)[] pages) => PDFDocument(context, fileName, pages, null);
01373
01374 /// <summary>
01375 /// Create a new PDF document containing the specified (parts of) pages from other documents.
01376 /// </summary>
01377 /// <param name="context">The context that was used to open the documents.</param>
01378 /// <param name="fileName">The output file name.</param>
01379 /// <param name="pages">The pages to include in the document.</param>
01380 /// <param name="options">Options for the output format.</param>
01381     public static void PDFDocument(MuPDFContext context, string fileName,
      IEnumerable<MuPDFPage> pages, PDFCreationOptions options = default) => PDFDocument(context, fileName,
      pages.Select(x => (x, x.Bounds, 1.0f)), options);
01382
01383 /// <summary>
01384 /// Create a new PDF document containing the specified (parts of) pages from other documents.
01385 /// </summary>
01386 /// <param name="context">The context that was used to open the documents.</param>
01387 /// <param name="fileName">The output file name.</param>
01388 /// <param name="pages">The pages to include in the document.</param>
01389     public static void PDFDocument(MuPDFContext context, string fileName, params MuPDFPage[]
      pages) => PDFDocument(context, fileName, pages, null);
01390
01391 /// <summary>
01392 /// Create a new SVG document containing the specified (parts of) pages from other documents.
01393 /// </summary>
01394 /// <param name="context">The context that was used to open the documents.</param>
01395 /// <param name="fileName">The output file name.</param>
01396 /// <param name="page">The page to include in the document.</param>
01397 /// <param name="region">The area of the page that should be included in the document</param>
01398 /// <param name="zoom">How much the region should be scaled.</param>
01399 /// <param name="options">Options for the output format.</param>
01400     public static void SVGDocument(MuPDFContext context, string fileName, MuPDFPage page,
      Rectangle region, float zoom, SVGCreationOptions options = default)
01401     {
01402         options = options ?? new SVGCreationOptions();
01403         string optionString = options.GetOptionString();
01404         CreateDocument(context, fileName, DocumentOutputFileTypes.SVG, new (MuPDFPage,
          Rectangle, float)[] { (page, region, zoom) }, optionString, options.IncludeAnnotations);
01405     }
01406
01407 /// <summary>
01408 /// Create a new SVG document containing the specified (parts of) pages from other documents.
01409 /// </summary>
01410 /// <param name="context">The context that was used to open the documents.</param>
01411 /// <param name="fileName">The output file name.</param>
01412 /// <param name="page">The page to include in the document.</param>
01413 /// <param name="options">Options for the output format.</param>

```

```

01414         public static void SVGDocument(MuPDFContext context, string fileName, MuPDFPage page,
01415             SVGCreationOptions options = default) => SVGDocument(context, fileName, page, page.Bounds, 1.0f,
01416             options);
01417     /// <summary>
01418     /// Create a new CBZ document containing the specified (parts of) pages from other documents.
01419     /// <param name="context">The context that was used to open the documents.</param>
01420     /// <param name="fileName">The output file name.</param>
01421     /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01422     /// the "region" element the area of the page that should be included in the document, and the "zoom"
01423     /// element how much the region should be scaled.</param>
01424     /// <param name="options">Options for the output format.</param>
01425         public static void CBZDocument(MuPDFContext context, string fileName,
01426             IEnumerable<(MuPDFPage page, Rectangle region)> pages, CBZCreationOptions options =
01427             default)
01428     {
01429         options = options ?? new CBZCreationOptions();
01430         string optionString = options.GetOptionString();
01431         CreateDocument(context, fileName, DocumentOutputFileTypes.CBZ, pages, optionString,
01432             options.IncludeAnnotations);
01433     }
01434     /// <summary>
01435     /// Create a new CBZ document containing the specified (parts of) pages from other documents.
01436     /// <param name="context">The context that was used to open the documents.</param>
01437     /// <param name="fileName">The output file name.</param>
01438     /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01439     /// the "region" element the area of the page that should be included in the document, and the "zoom"
01440     /// element how much the region should be scaled.</param>
01441         public static void CBZDocument(MuPDFContext context, string fileName, params (MuPDFPage
01442             page, Rectangle region, float zoom)[] pages) => CBZDocument(context, fileName, pages, null);
01443     /// <summary>
01444     /// Create a new CBZ document containing the specified (parts of) pages from other documents.
01445     /// <param name="context">The context that was used to open the documents.</param>
01446     /// <param name="fileName">The output file name.</param>
01447     /// <param name="pages">The pages to include in the document.</param>
01448         public static void CBZDocument(MuPDFContext context, string fileName, params MuPDFPage[]
01449             pages) => CBZDocument(context, fileName, pages, null);
01450     /// <summary>
01451     /// Create a new text document containing the specified (parts of) pages from other documents.
01452     /// <param name="context">The context that was used to open the documents.</param>
01453     /// <param name="fileName">The output file name.</param>
01454     /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01455     /// the "region" element the area of the page that should be included in the document, and the "zoom"
01456     /// element how much the region should be scaled.</param>
01457     /// <param name="options">Options for the output format.</param>
01458         public static void TextDocument(MuPDFContext context, string fileName,
01459             IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, TXTCreationOptions options =
01460             default)
01461     {
01462         options = options ?? new TXTCreationOptions();
01463         string optionString = options.GetOptionString();
01464         CreateDocument(context, fileName, DocumentOutputFileTypes.TXT, pages, optionString,
01465             options.IncludeAnnotations);
01466     }
01467     /// <summary>
01468     /// Create a new text document containing the specified (parts of) pages from other documents.
01469     /// <param name="context">The context that was used to open the documents.</param>
01470     /// <param name="fileName">The output file name.</param>
01471     /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01472     /// the "region" element the area of the page that should be included in the document, and the "zoom"
01473     /// element how much the region should be scaled.</param>
01474         public static void TextDocument(MuPDFContext context, string fileName, params (MuPDFPage
01475             page, Rectangle region, float zoom)[] pages) => TextDocument(context, fileName, pages, null);
01476     /// <summary>
01477     /// Create a new text document containing the specified (parts of) pages from other documents.
01478     /// </summary>
01479 
```

```
01480 /// <param name="context">The context that was used to open the documents.</param>
01481 /// <param name="fileName">The output file name.</param>
01482 /// <param name="pages">The pages to include in the document.</param>
01483 /// <param name="options">Options for the output format.</param>
01484     public static void TextDocument(MuPDFContext context, string fileName,
01485         IEnumerable<MuPDFPage> pages, TXTCreationOptions options = default) => TextDocument(context, fileName,
01486             pages.Select(x => (x, x.Bounds, 1.0f)), options);
01487 /// <summary>
01488 /// Create a new text document containing the specified (parts of) pages from other documents.
01489 /// </summary>
01490 /// <param name="context">The context that was used to open the documents.</param>
01491 /// <param name="fileName">The output file name.</param>
01492     public static void TextDocument(MuPDFContext context, string fileName, params MuPDFPage[]
01493         pages) => TextDocument(context, fileName, pages, null);
01494 /// <summary>
01495 /// Create a new structured text XML document containing the specified (parts of) pages from other
01496 /// documents.
01497 /// </summary>
01498 /// <param name="context">The context that was used to open the documents.</param>
01499 /// <param name="fileName">The output file name.</param>
01500 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01501 /// the "region" element the area of the page that should be included in the document, and the "zoom"
01502 /// element how much the region should be scaled.</param>
01503 /// <param name="options">Options for the output format.</param>
01504     public static void StructuredTextDocument(MuPDFContext context, string fileName,
01505         IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, TXTCreationOptions options =
01506             default)
01507     {
01508         options = options ?? new TXTCreationOptions();
01509         string optionString = options.GetOptionString();
01510         CreateDocument(context, fileName, DocumentOutputFileTypes.StructuredText, pages,
01511             optionString, options.IncludeAnnotations);
01512     }
01513 /// <summary>
01514 /// Create a new structured text XML document containing the specified (parts of) pages from other
01515 /// documents.
01516 /// </summary>
01517 /// Create a new structured text XML document containing the specified (parts of) pages from other
01518 /// documents.
01519 /// <param name="context">The context that was used to open the documents.</param>
01520 /// <param name="fileName">The output file name.</param>
01521 /// <param name="pages">The pages to include in the document.</param>
01522 /// <param name="options">Options for the output format.</param>
01523     public static void StructuredTextDocument(MuPDFContext context, string fileName,
01524         IEnumerable<MuPDFPage> pages, TXTCreationOptions options = default) => StructuredTextDocument(context,
01525             fileName, pages.Select(x => (x, x.Bounds, 1.0f)), options);
01526 /// <summary>
01527 /// Create a new structured text XML document containing the specified (parts of) pages from other
01528 /// documents.
01529 /// <param name="context">The context that was used to open the documents.</param>
01530 /// <param name="fileName">The output file name.</param>
01531     public static void StructuredTextDocument(MuPDFContext context, string fileName, params
01532         MuPDFPage[] pages) => StructuredTextDocument(context, fileName, pages, null);
01533 /// <summary>
01534 /// Create a new HTML document containing the specified (parts of) pages from other documents.
01535 /// </summary>
01536 /// <param name="context">The context that was used to open the documents.</param>
01537 /// <param name="fileName">The output file name.</param>
01538 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01539 /// the "region" element the area of the page that should be included in the document, and the "zoom"
01540 /// element how much the region should be scaled.</param>
01541     public static void HTMLDocument(MuPDFContext context, string fileName,
01542         IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, HTMLCreationOptions options =
01543             default)
01544     {
01545         options = options ?? new HTMLCreationOptions();
01546         string optionString = options.GetOptionString();
```

```

01544             CreateDocument(context, fileName, DocumentOutputFileTypes.HTML, pages, optionString,
01545                 options.IncludeAnnotations);
01546
01547 /// <summary>
01548 /// Create a new HTML document containing the specified (parts of) pages from other documents.
01549 /// </summary>
01550 /// <param name="context">The context that was used to open the documents.</param>
01551 /// <param name="fileName">The output file name.</param>
01552 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01553     the "region" element the area of the page that should be included in the document, and the "zoom"
01554     element how much the region should be scaled.</param>
01555         public static void HTMLDocument(MuPDFContext context, string fileName, params (MuPDFPage
01556             page, Rectangle region, float zoom)[] pages) => HTMLDocument(context, fileName, pages, null);
01557
01558 /// <summary>
01559 /// Create a new HTML document containing the specified (parts of) pages from other documents.
01560 /// </summary>
01561 /// <param name="context">The context that was used to open the documents.</param>
01562         public static void HTMLDocument(MuPDFContext context, string fileName,
01563             IEnumerable<MuPDFPage> pages, HTMLCreationOptions options = default) => HTMLDocument(context,
01564                 fileName, pages.Select(x => (x, x.Bounds, 1.0f)), options);
01565
01566 /// <summary>
01567 /// Create a new HTML document containing the specified (parts of) pages from other documents.
01568 /// </summary>
01569 /// <param name="context">The context that was used to open the documents.</param>
01570         public static void HTMLDocument(MuPDFContext context, string fileName, params MuPDFPage[]
01571             pages) => HTMLDocument(context, fileName, pages, null);
01572
01573 /// <summary>
01574 /// Create a new XHTML document containing the specified (parts of) pages from other documents.
01575 /// </summary>
01576 /// <param name="context">The context that was used to open the documents.</param>
01577 /// <param name="fileName">The output file name.</param>
01578 /// <param name="pages">The pages to include in the document. The "page" element specifies the page,
01579     the "region" element the area of the page that should be included in the document, and the "zoom"
01580     element how much the region should be scaled.</param>
01581         public static void XHTMLDocument(MuPDFContext context, string fileName,
01582             IEnumerable<(MuPDFPage page, Rectangle region, float zoom)> pages, HTMLCreationOptions options =
01583                 default)
01584             {
01585                 options = options ?? new HTMLCreationOptions();
01586                 string optionString = options.GetOptionString();
01587                 CreateDocument(context, fileName, DocumentOutputFileTypes.XHTML, pages, optionString,
01588                     options.IncludeAnnotations);
01589
01590 /// <summary>
01591 /// Create a new XHTML document containing the specified (parts of) pages from other documents.
01592 /// </summary>
01593 /// <param name="context">The context that was used to open the documents.</param>
01594         public static void XHTMLDocument(MuPDFContext context, string fileName, params (MuPDFPage
01595             page, Rectangle region, float zoom)[] pages) => XHTMLDocument(context, fileName, pages, null);
01596
01597 /// <summary>
01598 /// Create a new XHTML document containing the specified (parts of) pages from other documents.
01599 /// </summary>
01600 /// <param name="context">The context that was used to open the documents.</param>
01601         public static void XHTMLDocument(MuPDFContext context, string fileName,
01602             IEnumerable<MuPDFPage> pages, HTMLCreationOptions options = default) => XHTMLDocument(context,
01603                 fileName, pages.Select(x => (x, x.Bounds, 1.0f)), options);
01604
01605 /// <summary>
01606 /// Create a new XHTML document containing the specified (parts of) pages from other documents.
01607 /// </summary>
01608 /// <param name="context">The context that was used to open the documents.</param>
01609         public static void XHTMLDocument(MuPDFContext context, string fileName, params MuPDFPage[]
01610             pages) => XHTMLDocument(context, fileName, pages, null);
01611
01612 }

```

8.9 MuPDFDocument.cs

```
00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using MuPDFCore.StructuredText;
00019 using System;
00020 using System.IO;
00021 using System.Runtime.InteropServices;
00022 using System.Text;
00023 using System.Threading;
00024 using System.Threading.Tasks;
00025 using System.Linq;
00026
00027 namespace MuPDFCore
00028 {
00029 /// <summary>
00030 /// A wrapper over a MuPDF document object, which contains possibly multiple pages.
00031 /// </summary>
00032     public partial class MuPDFDocument : IDisposable
00033     {
00034     /// <summary>
00035     /// If the document is an image, the horizontal resolution of the image. Otherwise, 72.
00036     /// </summary>
00037         internal double ImageXRes = double.NaN;
00038
00039     /// <summary>
00040     /// If the document is an image, the vertical resolution of the image. Otherwise, 72.
00041     /// </summary>
00042         internal double ImageYRes = double.NaN;
00043
00044     /// <summary>
00045     /// File extensions corresponding to the supported input formats.
00046     /// </summary>
00047         private static readonly string[] FileTypeMagics = new[]
00048     {
00049         ".pdf",
00050         ".xps",
00051         ".cbz",
00052         ".png",
00053         ".jpg",
00054         ".bmp",
00055         ".gif",
00056         ".tif",
00057         ".pnm",
00058         ".pam",
00059         ".epub",
00060         ".fb2",
00061         ".mobi",
00062         ".html",
00063         ".txt",
00064         ".docx",
00065         ".pptx",
00066         ".xlsx",
00067     };
00068
00069     /// <summary>
00070     /// An <see cref="IDisposable"/> with a value of null.
00071     /// </summary>
00072         private static IDisposable NullDataHolder = null;
00073
00074     /// <summary>
00075     /// The context that owns this document.
00076     /// </summary>
00077         internal readonly MuPDFContext OwnerContext;
00078
00079     /// <summary>
00080     /// A pointer to the native document object.
00081     /// </summary>
00082         internal readonly IntPtr NativeDocument;
00083
00084     /// <summary>
00085     /// A pointer to a native document object, cast to the PDF type.
```

```

00086 /// This may be <see cref="IntPtr.Zero"/> if the document is not a PDF document.
00087 /// </summary>
00088     internal readonly IntPtr NativePDFDocument;
00089
00090 /// <summary>
00091 /// A pointer to the native stream that was used to create this document (if any).
00092 /// </summary>
00093     private readonly IntPtr NativeStream = IntPtr.Zero;
00094
00095 /// <summary>
00096 /// The number of pages in the document.
00097 /// </summary>
00098     private int PageCount;
00099
00100 /// <summary>
00101 /// An <see cref="IDisposable"/> that will be disposed together with this object.
00102 /// </summary>
00103     private readonly IDisposable DataHolder = null;
00104
00105 /// <summary>
00106 /// A <see cref="GCHandle"/> that will be freed when this object is disposed.
00107 /// </summary>
00108     private GCHandle? DataHandle = null;
00109
00110 /// <summary>
00111 /// An array of <see cref="MuPDFDisplayList"/>, one for each page in the document.
00112 /// </summary>
00113     internal MuPDFDisplayList[] DisplayLists;
00114
00115 /// <summary>
00116 /// The pages contained in the document.
00117 /// </summary>
00118     public MuPDFPageCollection Pages { get; private set; }
00119
00120 /// <summary>
00121 /// Defines whether the images resulting from rendering operations should be clipped to the page
00122 /// boundaries.
00123     public bool ClipToPageBounds { get; set; } = true;
00124
00125 /// <summary>
00126 /// Describes the encryption state of the document.
00127 /// </summary>
00128     public EncryptionState EncryptionState { get; private set; }
00129
00130 /// <summary>
00131 /// Describes the restriction state of the document.
00132 /// </summary>
00133     public RestrictionState RestrictionState { get; private set; }
00134
00135 /// <summary>
00136 /// Describes the operations that are restricted on the document. This is not actually enforced by
00137 /// the library,
00138 /// but library users should only allow these operations if the document has been unlocked with the
00139 /// owner password
00140     public DocumentRestrictions Restrictions { get; private set; }
00141
00142     private MuPDFOutline _outline = null;
00143
00144 /// The document outline (table of contents). If this document does not have an outline, this object
00145 /// will be
00146 /// empty, but not null. The outline is loaded from the document at the first access.
00147     public MuPDFOutline Outline
00148     {
00149         get
00150         {
00151             if (this._outline == null)
00152             {
00153                 this._outline = new MuPDFOutline(this.OwnerContext, this);
00154             }
00155
00156             return this._outline;
00157         }
00158     }
00159
00160     private bool _optionalContentGroupsDataLoaded = false;
00161     private MuPDFOptionalContentGroupData _optionalContentGroupData = null;
00162
00163 /// <summary>
00164 /// Contains information about optional content groups (aka layers). If this document does not
00165 /// contain any optional content groups, this object will be null. The optional content group
00166 /// data is loaded from the document at the first access.
00167 /// </summary>
00168     public MuPDFOptionalContentGroupData OptionalContentGroupData

```

```

00169         {
00170             get
00171             {
00172                 if (!OptionalContentGroupsDataLoaded)
00173                 {
00174                     _optionalContentGroupData = MuPDFOptionalContentGroupData.Load(this);
00175                     _optionalContentGroupsDataLoaded = true;
00176                 }
00177
00178                 return _optionalContentGroupData;
00179             }
00180         }
00181
00182     /// <summary>
00183     /// Create a new <see cref="MuPDFDocument"/> from data bytes accessible through the specified pointer.
00184     /// </summary>
00185     /// <param name="context">The context that will own this document.</param>
00186     /// <param name="dataAddress">A pointer to the data bytes that make up the document.</param>
00187     /// <param name="dataLength">The number of bytes to read from the specified address.</param>
00188     /// <param name="fileType">The type of the document to read.</param>
00189     public MuPDFDocument(MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes
00190     fileType) : this(context, dataAddress, dataLength, fileType, ref NullDataHolder) { }
00191
00192     /// <summary>
00193     /// Create a new <see cref="MuPDFDocument"/> from data bytes accessible through the specified pointer.
00194     /// </summary>
00195     /// <param name="context">The context that will own this document.</param>
00196     /// <param name="dataAddress">A pointer to the data bytes that make up the document.</param>
00197     /// <param name="dataLength">The number of bytes to read from the specified address.</param>
00198     /// <param name="dataHolder">An <see cref="IDisposable"/> that will be disposed when the <see
00199     /// cref="MuPDFDocument"/> is disposed.</param>
00200     public MuPDFDocument(MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes
00201     fileType, ref IDisposable dataHolder)
00202     {
00203         bool isImage = fileType == InputFileTypes.BMP || fileType == InputFileTypes.GIF ||
00204         fileType == InputFileTypes.JPG || fileType == InputFileTypes.PAM || fileType == InputFileTypes.PNG ||
00205         fileType == InputFileTypes.PNG || fileType == InputFileTypes.TIFF;
00206
00207         this.OwnerContext = context;
00208
00209         float xRes = 0;
00210         float yRes = 0;
00211
00212         ExitCodes result =
00213             (ExitCodes)NativeMethods.CreateDocumentFromStream(context.NativeContext, dataAddress,
00214             (ulong)dataLength, FileTypeMagics[(int)fileType], isImage ? 1 : 0, ref NativeDocument, ref
00215             NativeStream, ref PageCount, ref xRes, ref yRes);
00216
00217         if (xRes > 72)
00218         {
00219             this.ImageXRes = xRes;
00220         }
00221         else
00222         {
00223             this.ImageXRes = 72;
00224         }
00225
00226         if (yRes > 72)
00227         {
00228             this.ImageYRes = yRes;
00229         }
00230         else
00231         {
00232             this.ImageYRes = 72;
00233         }
00234
00235         this.DataHolder = dataHolder;
00236
00237         switch (result)
00238         {
00239             case ExitCodes.EXIT_SUCCESS:
00240                 break;
00241             case ExitCodes.ERR_CANNOT_OPEN_STREAM:
00242                 throw new MuPDFException("Cannot open data stream", result);
00243             case ExitCodes.ERR_CANNOT_OPEN_FILE:
00244                 throw new MuPDFException("Cannot open document", result);
00245             case ExitCodes.ERR_CANNOT_COUNT_PAGES:
00246                 throw new MuPDFException("Cannot count pages", result);
00247             default:
00248                 throw new MuPDFException("Unknown error", result);
00249         }
00250
00251         if (NativeMethods.CheckIfPasswordNeeded(context.NativeContext, this.NativeDocument) != 0)
00252         {
00253             this.EncryptionState = EncryptionState.Encrypted;
00254         }

```

```

00248         else
00249         {
00250             this.EncryptionState = EncryptionState.Unencrypted;
00251         }
00252
00253     int permissions = NativeMethods.GetPermissions(context.NativeContext,
00254     this.NativeDocument);
00255
00256     int restrictions = 0;
00257
00258     if ((permissions & 1) == 0)
00259     {
00260         restrictions |= 1;
00261     }
00262
00263     if ((permissions & 2) == 0)
00264     {
00265         restrictions |= 2;
00266     }
00267
00268     if ((permissions & 4) == 0)
00269     {
00270         restrictions |= 4;
00271     }
00272
00273     if ((permissions & 8) == 0)
00274     {
00275         restrictions |= 8;
00276     }
00277
00278     if (restrictions == 0)
00279     {
00280         this.Restrictions = DocumentRestrictions.None;
00281         this.RestrictionState = RestrictionState.Unrestricted;
00282     }
00283     else
00284     {
00285         this.Restrictions = (DocumentRestrictions)restrictions;
00286         this.RestrictionState = RestrictionState.Restricted;
00287     }
00288
00289     Pages = new MuPDFPageCollection(context, this, PageCount);
00290     DisplayLists = new MuPDFDisplayList[PageCount];
00291
00292     IntPtr pdfDocument = IntPtr.Zero;
00293     result = (ExitCodes)NativeMethods.GetPDFDocument(context.NativeContext,
00294     this.NativeDocument, ref pdfDocument);
00295
00296     switch (result)
00297     {
00298         case ExitCodes.EXIT_SUCCESS:
00299             this.NativePDFDocument = pdfDocument;
00300             break;
00301         case ExitCodes.ERR_CANNOT_CONVERT_TO_PDF:
00302             this.NativePDFDocument = IntPtr.Zero;
00303             break;
00304         default:
00305             throw new MuPDFException("Unknown error", result);
00306     }
00307 /// <summary>
00308 /// Create a new <see cref="MuPDFDocument"/> from an array of bytes.
00309 /// </summary>
00310 /// <param name="context">The context that will own this document.</param>
00311 /// <param name="data">An array containing the data bytes that make up the document. This must not be
00312 /// altered until after the <see cref="MuPDFDocument"/> has been disposed!
00313 /// The address of the array will be pinned, which may cause degradation in the Garbage Collector's
00314 /// performance, and is thus only advised for short-lived documents. To avoid this issue, marshal the
00315 /// bytes to unmanaged memory and use one of the <see cref="IntPtr"/> constructors.</param>
00316 /// <param name="fileType">The type of the document to read.</param>
00317     public MuPDFDocument(MuPDFContext context, byte[] data, InputFileTypes fileType)
00318     {
00319         bool isImage = fileType == InputFileTypes.BMP || fileType == InputFileTypes.GIF ||
00320         fileType == InputFileTypes.JPG || fileType == InputFileTypes.PAM || fileType == InputFileTypes.PNG ||
00321         fileType == InputFileTypes.PNG || fileType == InputFileTypes.TIFF;
00322
00323         this.OwnerContext = context;
00324
00325         DataHandle = GCHandle.Alloc(data, GCHandleType.Pinned);
00326         IntPtr dataAddress = DataHandle.Value.AddrOfPinnedObject();
00327         ulong dataLength = (ulong)data.Length;
00328
00329         float xRes = 0;
00330         float yRes = 0;
00331
00332         ExitCodes result =

```

```
(ExitCodes)NativeMethods.CreateDocumentFromStream(context.NativeContext, dataAddress, dataLength,
FileTypeMagics[(int)fileType], isImage ? 1 : 0, ref NativeDocument, ref PageCount,
ref xRes, ref yRes);
00328
00329     if (xRes > 72)
00330     {
00331         this.ImageXRes = xRes;
00332     }
00333     else
00334     {
00335         this.ImageXRes = 72;
00336     }
00337
00338     if (yRes > 72)
00339     {
00340         this.ImageYRes = yRes;
00341     }
00342     else
00343     {
00344         this.ImageYRes = 72;
00345     }
00346
00347     switch (result)
00348     {
00349         case ExitCodes.EXIT_SUCCESS:
00350             break;
00351         case ExitCodes.ERR_CANNOT_OPEN_STREAM:
00352             throw new MuPDFException("Cannot open data stream", result);
00353         case ExitCodes.ERR_CANNOT_OPEN_FILE:
00354             throw new MuPDFException("Cannot open document", result);
00355         case ExitCodes.ERR_CANNOT_COUNT_PAGES:
00356             throw new MuPDFException("Cannot count pages", result);
00357         default:
00358             throw new MuPDFException("Unknown error", result);
00359     }
00360
00361     if (NativeMethods.CheckIfPasswordNeeded(context.NativeContext, this.NativeDocument) != 0)
00362     {
00363         this.EncryptionState = EncryptionState.Encrypted;
00364     }
00365     else
00366     {
00367         this.EncryptionState = EncryptionState.Unencrypted;
00368     }
00369
00370     int permissions = NativeMethods.GetPermissions(context.NativeContext,
this.NativeDocument);
00371
00372     int restrictions = 0;
00373
00374     if ((permissions & 1) == 0)
00375     {
00376         restrictions |= 1;
00377     }
00378
00379     if ((permissions & 2) == 0)
00380     {
00381         restrictions |= 2;
00382     }
00383
00384     if ((permissions & 4) == 0)
00385     {
00386         restrictions |= 4;
00387     }
00388
00389     if ((permissions & 8) == 0)
00390     {
00391         restrictions |= 8;
00392     }
00393
00394     if (restrictions == 0)
00395     {
00396         this.Restrictions = DocumentRestrictions.None;
00397         this.RestrictionState = RestrictionState.Unrestricted;
00398     }
00399     else
00400     {
00401         this.Restrictions = (DocumentRestrictions)restrictions;
00402         this.RestrictionState = RestrictionState.Restricted;
00403     }
00404
00405     Pages = new MuPDFPageCollection(context, this, PageCount);
00406     DisplayLists = new MuPDFDisplayList[PageCount];
00407
00408     IntPtr pdfDocument = IntPtr.Zero;
00409     result = (ExitCodes)NativeMethods.GetPDFDocument(context.NativeContext,
this.NativeDocument, ref pdfDocument);
```

```

00410
00411         switch (result)
00412     {
00413         case ExitCodes.EXIT_SUCCESS:
00414             this.NativePDFDocument = pdfDocument;
00415             break;
00416         case ExitCodes.ERR_CANNOT_CONVERT_TO_PDF:
00417             this.NativePDFDocument = IntPtr.Zero;
00418             break;
00419         default:
00420             throw new MuPDFException("Unknown error", result);
00421     }
00422 }
00423
00424 /// <summary>
00425 /// Create a new <see cref="MuPDFDocument"/> from a <see cref="MemoryStream"/>.
00426 /// </summary>
00427 /// <param name="context">The context that will own this document.</param>
00428 /// <param name="data">The <see cref="MemoryStream"/> containing the data that makes up the document.
00429 This will be disposed when the <see cref="MuPDFDocument"/> has been disposed and must not be disposed
00430 externally!
00431 /// The address of the <see cref="MemoryStream"/>'s buffer will be pinned, which may cause degradation
00432 in the Garbage Collector's performance, and is thus only advised for short-lived documents. To avoid
00433 this issue, marshal the bytes to unmanaged memory and use one of the <see cref="IntPtr"/>
00434 constructors.</param>
00435 /// <param name="fileType">The type of the document to read.</param>
00436 public MuPDFDocument(MuPDFContext context, ref MemoryStream data, InputFileTypes fileType)
00437 {
00438     bool isImage = fileType == InputFileTypes.BMP || fileType == InputFileTypes.GIF ||
00439     fileType == InputFileTypes.JPG || fileType == InputFileTypes.PAM || fileType == InputFileTypes.PNG ||
00440     fileType == InputFileTypes.PNG || fileType == InputFileTypes.TIFF;
00441
00442     this.OwnerContext = context;
00443
00444     int origin = (int)data.Seek(0, SeekOrigin.Begin);
00445     ulong dataLength = (ulong)data.Length;
00446     byte[] dataBytes = data.GetBuffer();
00447
00448     DataHandle = GCHandle.Alloc(dataBytes, GCHandleType.Pinned);
00449     IntPtr dataAddress = IntPtr.Add(DataHandle.Value.AddrOfPinnedObject(), origin);
00450
00451     DataHolder = data;
00452
00453     float xRes = 0;
00454     float yRes = 0;
00455
00456     ExitCodes result =
00457     (ExitCodes)NativeMethods.CreateDocumentFromStream(context.NativeContext, dataAddress, dataLength,
00458     FileTypeMagics[(int)fileType], isImage ? 1 : 0, ref NativeDocument, ref NativeStream, ref PageCount,
00459     ref xRes, ref yRes);
00460
00461     if (xRes > 72)
00462     {
00463         this.ImageXRes = xRes;
00464     }
00465     else
00466     {
00467         this.ImageXRes = 72;
00468     }
00469
00470     if (yRes > 72)
00471     {
00472         this.ImageYRes = yRes;
00473     }
00474     else
00475     {
00476         this.ImageYRes = 72;
00477     }
00478
00479     switch (result)
00480     {
00481         case ExitCodes.EXIT_SUCCESS:
00482             break;
00483         case ExitCodes.ERR_CANNOT_OPEN_STREAM:
00484             throw new MuPDFException("Cannot open data stream", result);
00485         case ExitCodes.ERR_CANNOT_OPEN_FILE:
00486             throw new MuPDFException("Cannot open document", result);
00487         case ExitCodes.ERR_CANNOT_COUNT_PAGES:
00488             throw new MuPDFException("Cannot count pages", result);
00489         default:
00490             throw new MuPDFException("Unknown error", result);
00491     }
00492
00493     if (NativeMethods.CheckIfPasswordNeeded(context.NativeContext, this.NativeDocument) != 0)
00494     {
00495         this.EncryptionState = EncryptionState.Encrypted;
00496     }

```

```
00487         }
00488     else
00489     {
00490         this.EncryptionState = EncryptionState.Unencrypted;
00491     }
00492 
00493     int permissions = NativeMethods.GetPermissions(context.NativeContext,
00494         this.NativeDocument);
00495 
00496     int restrictions = 0;
00497 
00498     if ((permissions & 1) == 0)
00499     {
00500         restrictions |= 1;
00501     }
00502 
00503     if ((permissions & 2) == 0)
00504     {
00505         restrictions |= 2;
00506     }
00507 
00508     if ((permissions & 4) == 0)
00509     {
00510         restrictions |= 4;
00511     }
00512 
00513     if ((permissions & 8) == 0)
00514     {
00515         restrictions |= 8;
00516     }
00517 
00518     if (restrictions == 0)
00519     {
00520         this.Restrictions = DocumentRestrictions.None;
00521         this.RestrictionState = RestrictionState.Unrestricted;
00522     }
00523     else
00524     {
00525         this.Restrictions = (DocumentRestrictions)restrictions;
00526         this.RestrictionState = RestrictionState.Restricted;
00527     }
00528 
00529     Pages = new MuPDFPageCollection(context, this, PageCount);
00530     DisplayLists = new MuPDFDisplayList[PageCount];
00531 
00532     IntPtr pdfDocument = IntPtr.Zero;
00533     result = (ExitCodes)NativeMethods.GetPDFDocument(context.NativeContext,
00534         this.NativeDocument, ref pdfDocument);
00535 
00536     switch (result)
00537     {
00538         case ExitCodes.EXIT_SUCCESS:
00539             this.NativePDFDocument = pdfDocument;
00540             break;
00541         case ExitCodes.ERR_CANNOT_CONVERT_TO_PDF:
00542             this.NativePDFDocument = IntPtr.Zero;
00543             break;
00544         default:
00545             throw new MuPDFException("Unknown error", result);
00546     }
00547 /// <summary>
00548 /// Create a new <see cref="MuPDFDocument"/> from a file.
00549 /// </summary>
00550 /// <param name="context">The context that will own this document.</param>
00551 /// <param name="fileName">The path to the file to open.</param>
00552     public MuPDFDocument(MuPDFContext context, string fileName)
00553     {
00554         bool isImage;
00555 
00556         string extension = Path.GetExtension(fileName).ToLowerInvariant();
00557 
00558         switch (extension)
00559         {
00560             case ".bmp":
00561             case ".dib":
00562 
00563             case ".gif":
00564 
00565             case ".jpg":
00566             case ".jpeg":
00567             case ".jpe":
00568             case ".jfif":
00569             case ".jfif":
00570             case ".jfi":
```

```
00572         case ".pam":  
00573         case ".pbm":  
00574         case ".pgm":  
00575         case ".ppm":  
00576         case ".pnm":  
00577             case ".png":  
00578             case ".tif":  
00579             case ".tiff":  
00580                 isImage = true;  
00581                 break;  
00582             default:  
00583                 isImage = false;  
00584                 break;  
00585         }  
00586  
00587     }  
00588  
00589     this.OwnerContext = context;  
00590  
00591     float xRes = 0;  
00592     float yRes = 0;  
00593  
00594     ExitCodes result;  
00595  
00596     using (UTF8EncodedString encodedFileName = new UTF8EncodedString(fileName))  
00597     {  
00598         result = (ExitCodes)NativeMethods.CreateDocumentFromFile(context.NativeContext,  
encodedFileName.Address, isImage ? 1 : 0, ref NativeDocument, ref PageCount, ref xRes, ref yRes);  
00599     }  
00600  
00601     if (xRes > 72)  
00602     {  
00603         this.ImageXRes = xRes;  
00604     }  
00605     else  
00606     {  
00607         this.ImageXRes = 72;  
00608     }  
00609  
00610     if (yRes > 72)  
00611     {  
00612         this.ImageYRes = yRes;  
00613     }  
00614     else  
00615     {  
00616         this.ImageYRes = 72;  
00617     }  
00618  
00619     switch (result)  
00620     {  
00621         case ExitCodes.EXIT_SUCCESS:  
00622             break;  
00623         case ExitCodes.ERR_CANNOT_OPEN_FILE:  
00624             throw new MuPDFException("Cannot open document", result);  
00625         case ExitCodes.ERR_CANNOT_COUNT_PAGES:  
00626             throw new MuPDFException("Cannot count pages", result);  
00627         default:  
00628             throw new MuPDFException("Unknown error", result);  
00629     }  
00630  
00631     if (NativeMethods.CheckIfPasswordNeeded(context.NativeContext, this.NativeDocument) != 0)  
00632     {  
00633         this.EncryptionState = EncryptionState.Encrypted;  
00634     }  
00635     else  
00636     {  
00637         this.EncryptionState = EncryptionState.Unencrypted;  
00638     }  
00639  
00640     int permissions = NativeMethods.GetPermissions(context.NativeContext,  
this.NativeDocument);  
00641  
00642     int restrictions = 0;  
00643  
00644     if ((permissions & 1) == 0)  
00645     {  
00646         restrictions |= 1;  
00647     }  
00648  
00649     if ((permissions & 2) == 0)  
00650     {  
00651         restrictions |= 2;  
00652     }  
00653  
00654     if ((permissions & 4) == 0)  
00655     {  
00656
```

```

00657             restrictions |= 4;
00658         }
00659
00660         if ((permissions & 8) == 0)
00661     {
00662         restrictions |= 8;
00663     }
00664
00665         if (restrictions == 0)
00666     {
00667         this.Restrictions = DocumentRestrictions.None;
00668         this.RestrictionState = RestrictionState.Unrestricted;
00669     }
00670     else
00671     {
00672         this.Restrictions = (DocumentRestrictions)restrictions;
00673         this.RestrictionState = RestrictionState.Restricted;
00674     }
00675
00676     Pages = new MuPDFPageCollection(context, this, PageCount);
00677     DisplayLists = new MuPDFDisplayList[PageCount];
00678
00679     IntPtr pdfDocument = IntPtr.Zero;
00680     result = (ExitCodes)NativeMethods.GetPDFDocument(context.NativeContext,
00681     this.NativeDocument, ref pdfDocument);
00682
00683     switch (result)
00684     {
00685         case ExitCodes.EXIT_SUCCESS:
00686             this.NativePDFDocument = pdfDocument;
00687             break;
00688         case ExitCodes.ERR_CANNOT_CONVERT_TO_PDF:
00689             this.NativePDFDocument = IntPtr.Zero;
00690             break;
00691         default:
00692             throw new MuPDFException("Unknown error", result);
00693     }
00694
00695 /// <summary>
00696 /// Discard all the display lists that have been loaded from the document, possibly freeing some
00697 /// memory in the case of a huge document.
00698 /// </summary>
00699     public void ClearCache()
00700     {
00701         for (int i = 0; i < PageCount; i++)
00702         {
00703             DisplayLists[i]?.Dispose();
00704             DisplayLists[i] = null;
00705         }
00706
00707 /// <summary>
00708 /// Sets the document layout for reflowable document types (e.g., HTML, MOBI). Does not have any
00709 /// effect for documents with a fixed layout (e.g., PDF).
00710 /// <param name="width">The width of each page, in points. Must be > 0.</param>
00711 /// <param name="height">The height of each page, in points. Must be > 0.</param>
00712 /// <param name="em">The default font size, in points.</param>
00713     public void Layout(float width, float height, float em)
00714     {
00715         if (width <= 0)
00716         {
00717             throw new ArgumentOutOfRangeException(nameof(width), width, "The page width must be
00718             greater than 0!");
00719         }
00720         if (height <= 0)
00721         {
00722             throw new ArgumentOutOfRangeException(nameof(height), height, "The page height must be
00723             greater than 0!");
00724
00725             this.ClearCache();
00726             this.Pages.Dispose();
00727
00728             NativeMethods.LayoutDocument(this.OwnerContext.NativeContext, this.NativeDocument, width,
00729             height, em, out int pageCount);
00730
00731             this.PageCount = pageCount;
00732             this.Pages = new MuPDFPageCollection(this.OwnerContext, this, PageCount);
00733             this.DisplayLists = new MuPDFDisplayList[PageCount];
00734         }
00735 /// <summary>
00736 /// Sets the document layout for reflowable document types (e.g., HTML, MOBI), so that the document is
00737 /// rendered to a single

```

```

00737 /// page, as tall as necessary. Does not have any effect for documents with a fixed layout (e.g.,
00738 /// PDF).
00739 /// </summary>
00740 /// <param name="width">The width of the page, in points. Must be &gt; 0.</param>
00741     public void LayoutSinglePage(float width, float em)
00742     {
00743         if (width <= 0)
00744         {
00745             throw new ArgumentOutOfRangeException(nameof(width), width, "The page width must be
00746 greater than 0!");
00747         }
00748         this.ClearCache();
00749         this.Pages.Dispose();
00750
00751         NativeMethods.LayoutDocument(this.OwnerContext.NativeContext, this.NativeDocument, width,
00752         0, em, out int pageCount);
00753
00754         this.PageCount = pageCount;
00755         this.Pages = new MuPDFPageCollection(this.OwnerContext, this, PageCount);
00756         this.DisplayLists = new MuPDFDisplayList[PageCount];
00757     }
00758 /// <summary>
00759 /// Render (part of) a page to an array of bytes.
00760 /// </summary>
00761 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
00762 /// <param name="region">The region of the page to render in page units.</param>
00763 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
00764 /// pixel of the image.</param>
00765 /// <param name="pixelFormat">The format of the pixel data.</param>
00766 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00767 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
00768 /// included.</param>
00769 /// <returns>A byte array containing the raw values for the pixels of the rendered image.</returns>
00770     public byte[] Render(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat,
00771     bool includeAnnotations = true)
00772     {
00773         if (this.EncryptionState == EncryptionState.Encrypted)
00774         {
00775             throw new DocumentLockedException("A password is necessary to render the document!");
00776         }
00777
00778         int bufferSize = MuPDFDocument.GetRenderedSize(region, zoom, pixelFormat);
00779
00780         byte[] buffer = new byte[bufferSize];
00781
00782         GCHandle bufferHandle = GCHandle.Alloc(buffer, GCHandleType.Pinned);
00783         IntPtr bufferPointer = bufferHandle.AddrOfPinnedObject();
00784
00785         try
00786         {
00787             Render(pageNumber, region, zoom, pixelFormat, bufferPointer, includeAnnotations);
00788         }
00789         finally
00790         {
00791             bufferHandle.Free();
00792         }
00793     }
00794 /// <summary>
00795 /// Render a page to an array of bytes.
00796 /// </summary>
00797 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
00798 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
00799 /// pixel of the image.</param>
00800 /// <param name="pixelFormat">The format of the pixel data.</param>
00801 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00802 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
00803 /// included.</param>
00804 /// <returns>A byte array containing the raw values for the pixels of the rendered image.</returns>
00805     public byte[] Render(int pageNumber, double zoom, PixelFormats pixelFormat, bool
00806     includeAnnotations = true)
00807     {
00808         if (this.EncryptionState == EncryptionState.Encrypted)
00809         {
00810             throw new DocumentLockedException("A password is necessary to render the document!");
00811         }
00812     }

```

```
00813 /// Render (part of) a page to the specified destination.
00814 /// </summary>
00815 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
00816 /// <param name="region">The region of the page to render in page units.</param>
00817 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
00818 /// pixel of the image.</param>
00819 /// <param name="pixelFormat">The format of the pixel data.</param>
00820 /// <param name="destination">The address of the buffer where the pixel data will be written. There
00821 /// must be enough space available to write the values for all the pixels, otherwise this will fail
00822 /// catastrophically!</param>
00823 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00824 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
00825 /// included.</param>
00826     public void Render(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat,
00827     IntPtr destination, bool includeAnnotations = true)
00828     {
00829         if (this.EncryptionState == EncryptionState.Encrypted)
00830         {
00831             throw new DocumentLockedException("A password is necessary to render the document!");
00832         }
00833         if (DisplayLists[pageNumber] == null)
00834         {
00835             DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
00836             this.Pages[pageNumber], includeAnnotations);
00837         }
00838         if (zoom < 0.000001 || zoom * region.Width <= 0.001 || zoom * region.Height <= 0.001)
00839         {
00840             throw new ArgumentOutOfRangeException(nameof(zoom), zoom, "The zoom factor is too
00841             small!");
00842         }
00843         if (this.ImageXRes != 72 || this.ImageYRes != 72)
00844         {
00845             zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
00846             region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 / this.ImageYRes,
00847             region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
00848             float fzoom = (float)zoom;
00849             ExitCodes result =
00850             (ExitCodes)NativeMethods.RenderSubDisplayList(OwnerContext.NativeContext,
00851             DisplayLists[pageNumber].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, fzoom,
00852             (int)pixelFormat, destination, IntPtr.Zero);
00853             switch (result)
00854             {
00855                 case ExitCodes.EXIT_SUCCESS:
00856                     break;
00857                 case ExitCodes.ERR_CANNOT_RENDER:
00858                     throw new MuPDFException("Cannot render page", result);
00859                 default:
00860                     throw new MuPDFException("Unknown error", result);
00861             }
00862             RoundedRectangle roundedRegion = region.Round(fzoom);
00863             RoundedSize roundedSize = new RoundedSize(roundedRegion.Width, roundedRegion.Height);
00864             if (pixelFormat == PixelFormats.RGBA || pixelFormat == PixelFormats.BGRA)
00865             {
00866                 Utils.UnpremultiplyAlpha(destination, roundedSize);
00867             }
00868             if (this.ClipToPageBounds &&
00869             !Pages[pageNumber].Bounds.Contains(DisplayLists[pageNumber].Bounds.Intersect(region)))
00870             {
00871                 Utils.ClipImage(destination, roundedSize, region, Pages[pageNumber].Bounds,
00872                 pixelFormat);
00873             }
00874         }
00875     /// <summary>
00876     /// Render a page to the specified destination.
00877     /// </summary>
00878     /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
00879     /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
00880     /// pixel of the image.</param>
00881     /// <param name="pixelFormat">The format of the pixel data.</param>
00882     /// <param name="destination">The address of the buffer where the pixel data will be written. There
00883     /// must be enough space available to write the values for all the pixels, otherwise this will fail
00884     /// catastrophically!</param>
00885     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
00886     /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
00887     /// included.</param>
00888     public void Render(int pageNumber, double zoom, PixelFormats pixelFormat, IntPtr destination,
```

```

0081     bool includeAnnotations = true)
0082     {
0083         if (this.EncryptionState == EncryptionState.Encrypted)
0084         {
0085             throw new DocumentLockedException("A password is necessary to render the document!");
0086         }
0087         Rectangle region = this.Pages[pageNumber].Bounds;
0088         Render(pageNumber, region, zoom, pixelFormat, destination, includeAnnotations);
0089     }
0090
0091 /// <summary>
0092 /// Render (part of) a page to a <see cref="Span{T}">Span</see>&lt;;<see cref="byte"/>&gt;.
0093 /// </summary>
0094 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
0095 /// <param name="region">The region of the page to render in page units.</param>
0096 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
0097 /// pixel of the image.</param>
0098 /// <param name="pixelFormat">The format of the pixel data.</param>
0099 /// <param name="disposable">An <see cref="IDisposable"/> that can be used to free the memory where
0100 /// the image is stored. You should keep track of this and dispose it when you have finished working with
0101 /// the image.</param>
0102 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
0103 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
0104 /// included.</param>
0105     public Span<byte> Render(int pageNumber, Rectangle region, double zoom, PixelFormats
0106     pixelFormat, out IDisposable disposable, bool includeAnnotations = true)
0107     {
0108         if (this.EncryptionState == EncryptionState.Encrypted)
0109         {
0110             throw new DocumentLockedException("A password is necessary to render the document!");
0111         }
0112
0113         int dataSize = GetRenderedSize(region, zoom, pixelFormat);
0114
0115         IntPtr destination = Marshal.AllocHGlobal(dataSize);
0116         disposable = new DisposableIntPtr(destination, dataSize);
0117
0118         this.Render(pageNumber, region, zoom, pixelFormat, destination, includeAnnotations);
0119
0120         unsafe
0121         {
0122             return new Span<byte>((void*)destination, dataSize);
0123         }
0124     }
0125
0126 /// <summary>
0127 /// Render a page to a <see cref="Span{T}">Span</see>&lt;;<see cref="byte"/>&gt;.
0128 /// </summary>
0129 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
0130 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
0131 /// pixel of the image.</param>
0132 /// <param name="pixelFormat">The format of the pixel data.</param>
0133 /// <param name="disposable">An <see cref="IDisposable"/> that can be used to free the memory where
0134 /// the image is stored. You should keep track of this and dispose it when you have finished working with
0135 /// the image.</param>
0136 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
0137 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
0138 /// included.</param>
0139     public Span<byte> Render(int pageNumber, double zoom, PixelFormats pixelFormat, out
0140     IDisposable disposable, bool includeAnnotations = true)
0141     {
0142         if (this.EncryptionState == EncryptionState.Encrypted)
0143         {
0144             throw new DocumentLockedException("A password is necessary to render the document!");
0145         }
0146
0147         Rectangle region = this.Pages[pageNumber].Bounds;
0148         return Render(pageNumber, region, zoom, pixelFormat, out disposable, includeAnnotations);
0149     }
0150
0151 /// <summary>
0152 /// Create a new <see cref="MuPDFMultiThreadedPageRendererer"/> that renders the specified page with the
0153 /// specified number of threads.
0154 /// </summary>
0155 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
0156 /// <param name="threadCount">The number of threads to use. This must be factorisable using only
0157 /// powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <paramref name="threadCount"/>
0158 /// that satisfies this condition is used.</param>
0159 /// <returns>A <see cref="MuPDFMultiThreadedPageRendererer"/> that can be used to render the specified
0160 /// page with the specified number of threads.</returns>
0161 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
0162 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
0163 /// included.</param>
0164     public MuPDFMultiThreadedPageRendererer GetMultiThreadedRendererer(int pageNumber, int
0165     threadCount, bool includeAnnotations = true)

```

```

00948         {
00949             if (this.EncryptionState == EncryptionState.Encrypted)
00950             {
00951                 throw new DocumentLockedException("A password is necessary to render the document!");
00952             }
00953
00954             if (DisplayLists[pageNumber] == null)
00955             {
00956                 DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
00957                     this.Pages[pageNumber], includeAnnotations);
00958             }
00959
00960             return new MuPDFMultiThreadedPageRenderer(OwnerContext, DisplayLists[pageNumber],
00961                 threadCount, Pages[pageNumber].Bounds, this.ClipToPageBounds, this.ImageXRes, this.ImageYRes);
00962         }
00963
00964     /// <summary>
00965     /// Determine how many bytes will be necessary to render the specified page at the specified zoom
00966     /// level, using the the specified pixel format.
00967     /// </summary>
00968     /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
00969     /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
00970     /// pixel of the image.</param>
00971     /// <param name="pixelFormat">The format of the pixels data.</param>
00972     /// <returns>An integer representing the number of bytes that will be necessary to store the pixel
00973     /// data of the rendered image.</returns>
00974     public int GetRenderedSize(int pageNumber, double zoom, PixelFormats pixelFormat)
00975     {
00976         if (this.EncryptionState == EncryptionState.Encrypted)
00977         {
00978             throw new DocumentLockedException("A password is necessary to render the document!");
00979         }
00980
00981     /// <summary>
00982     /// Determine how many bytes will be necessary to render the specified region in page units at the
00983     /// specified zoom level, using the the specified pixel format.
00984     /// </summary>
00985     /// <param name="region">The region that will be rendered.</param>
00986     /// <param name="zoom">The scale at which the region will be rendered. This will determine the size
00987     /// in pixel of the image.</param>
00988     /// <param name="pixelFormat">The format of the pixels data.</param>
00989     /// <returns>An integer representing the number of bytes that will be necessary to store the pixel
00990     /// data of the rendered image.</returns>
00991     public static int GetRenderedSize(Rectangle region, double zoom, PixelFormats pixelFormat)
00992     {
00993         float x0 = region.X0 * (float)zoom;
00994         float y0 = region.Y0 * (float)zoom;
00995         float x1 = region.X1 * (float)zoom;
00996         float y1 = region.Y1 * (float)zoom;
00997
00998         Rectangle scaledRect = new Rectangle(x0, y0, x1, y1);
00999         RoundedRectangle bounds = scaledRect.Round();
01000
01001         int width = bounds.Width;
01002         int height = bounds.Height;
01003
01004         switch (pixelFormat)
01005         {
01006             case PixelFormats.RGB:
01007             case PixelFormats.BGR:
01008                 return width * height * 3;
01009             case PixelFormats.RGBA:
01010             case PixelFormats.BGRA:
01011                 return width * height * 4;
01012         }
01013
01014     /// <summary>
01015     /// Save (part of) a page to an image file in the specified format.
01016     /// </summary>
01017     /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01018     /// <param name="region">The region of the page to render in page units.</param>
01019     /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01020     /// pixel of the image.</param>
01021     /// <param name="pixelFormat">The format of the pixel data.</param>
01022     /// <param name="fileName">The path to the output file.</param>
01023     /// <param name="fileType">The output format of the file.</param>
01024     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01025     /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
01026     /// included.</param>
01027     public void SaveImage(int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat,
01028                     string fileName, RasterOutputFileTypes fileType, bool includeAnnotations = true)

```

```

01023         {
01024             if (this.EncryptionState == EncryptionState.Encrypted)
01025             {
01026                 throw new DocumentLockedException("A password is necessary to render the document!");
01027             }
01028
01029             if (pixelFormat == PixelFormats.RGBA && fileType == RasterOutputFileTypes.PNM)
01030             {
01031                 throw new ArgumentException("Cannot save an image with alpha channel in PNM format!",
01032                     nameof(fileType));
01033
01034             if (pixelFormat != PixelFormats.RGB && fileType == RasterOutputFileTypes.JPG)
01035             {
01036                 throw new ArgumentException("The JPEG format only supports RGB pixel data without an
01037                     alpha channel!", nameof(fileType));
01038
01039             if ((pixelFormat != PixelFormats.RGB && pixelFormat != PixelFormats.RGBA) && fileType ==
01040                 RasterOutputFileTypes.PNG)
01041             {
01042                 throw new ArgumentException("The PNG format only supports RGB or RGBA pixel data!",
01043                     nameof(fileType));
01044
01045             if (DisplayLists[pageNumber] == null)
01046             {
01047                 DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01048                     this.Pages[pageNumber], includeAnnotations);
01049
01050             if (zoom < 0.000001 || zoom * region.Width <= 0.001 || zoom * region.Height <= 0.001)
01051             {
01052                 throw new ArgumentOutOfRangeException(nameof(zoom), zoom, "The zoom factor is too
01053                     small!");
01054             }
01055
01056             if (this.ImageXRes != 72 || this.ImageYRes != 72)
01057             {
01058                 zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01059                 region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 /
01060                     this.ImageYRes, region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01061
01062             float fzoom = (float)zoom;
01063
01064             ExitCodes result;
01065
01066             using (UTF8EncodedString encodedFileName = new UTF8EncodedString(fileName))
01067             {
01068                 result = (ExitCodes)NativeMethods.SaveImage(OwnerContext.NativeContext,
01069                     DisplayLists[pageNumber].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, fzoom,
01070                     (int)pixelFormat, encodedFileName.Address, (int)fileType, 90);
01071
01072             switch (result)
01073             {
01074                 case ExitCodes.EXIT_SUCCESS:
01075                     break;
01076                 case ExitCodes.ERR_CANNOT_RENDER:
01077                     throw new MuPDFException("Cannot render page", result);
01078                 case ExitCodes.ERR_CANNOT_SAVE:
01079                     throw new MuPDFException("Cannot save to the output file", result);
01080                 default:
01081                     throw new MuPDFException("Unknown error", result);
01082             }
01083
01084 /// <summary>
01085 /// Save (part of) a page to an image file in JPEG format, with the specified quality.
01086 /// </summary>
01087 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01088 /// <param name="region">The region of the page to render in page units.</param>
01089 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01090 /// pixel of the image.</param>
01091 /// <param name="fileName">The path to the output file.</param>
01092 /// <param name="quality">The quality of the JPEG output file (ranging from 0 to 100).</param>
01093 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01094 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
01095 /// included.</param>
01096         public void SaveImageAsJPEG(int pageNumber, Rectangle region, double zoom, string fileName,
01097             int quality, bool includeAnnotations = true)
01098         {
01099             if (this.EncryptionState == EncryptionState.Encrypted)
01100             {
01101                 throw new DocumentLockedException("A password is necessary to render the document!");
01102             }

```

```

01097             if (quality < 0 || quality > 100)
01098             {
01099                 throw new ArgumentOutOfRangeException(nameof(quality), quality, "The JPEG quality must
01100                     range between 0 and 100 (inclusive)!");
01101             }
01102
01103             if (DisplayLists[pageNumber] == null)
01104             {
01105                 DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01106                     this.Pages[pageNumber], includeAnnotations);
01107             }
01108             if (zoom < 0.000001 || zoom * region.Width <= 0.001 || zoom * region.Height <= 0.001)
01109             {
01110                 throw new ArgumentOutOfRangeException(nameof(zoom), zoom, "The zoom factor is too
01111                     small!");
01112             }
01113             if (this.ImageXRes != 72 || this.ImageYRes != 72)
01114             {
01115                 zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01116                 region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 /
01117                     this.ImageYRes, region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01118             }
01119             float fzoom = (float)zoom;
01120
01121             ExitCodes result;
01122
01123             using (UTF8EncodedString encodedFileName = new UTF8EncodedString(fileName))
01124             {
01125                 result = (ExitCodes)NativeMethods.SaveImage(OwnerContext.NativeContext,
01126                     DisplayLists[pageNumber].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, fzoom,
01127                     (int)PixelFormats.RGB, encodedFileName.Address, (int)RasterOutputFileTypes.JPEG, quality);
01128             }
01129             switch (result)
01130             {
01131                 case ExitCodes.EXIT_SUCCESS:
01132                     break;
01133                 case ExitCodes.ERR_CANNOT_RENDER:
01134                     throw new MuPDFException("Cannot render page", result);
01135                 case ExitCodes.ERR_CANNOT_SAVE:
01136                     throw new MuPDFException("Cannot save to the output file", result);
01137                 default:
01138                     throw new MuPDFException("Unknown error", result);
01139             }
01140         }
01141 /// <summary>
01142 /// Save a page to an image file in the specified format.
01143 /// </summary>
01144 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01145 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01146 /// pixel of the image.</param>
01147 /// <param name="pixelFormat">The format of the pixel data.</param>
01148 /// <param name="fileName">The path to the output file.</param>
01149 /// <param name="fileType">The output format of the file.</param>
01150 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01151 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
01152 /// included.</param>
01153         public void SaveImage(int pageNumber, double zoom, PixelFormats pixelFormat, string fileName,
01154             RasterOutputFileTypes fileType, bool includeAnnotations = true)
01155         {
01156             if (this.EncryptionState == EncryptionState.Encrypted)
01157             {
01158                 throw new DocumentLockedException("A password is necessary to render the document!");
01159             }
01160
01161             Rectangle region = this.Pages[pageNumber].Bounds;
01162             SaveImage(pageNumber, region, zoom, pixelFormat, fileName, fileType, includeAnnotations);
01163         }
01164 /// <summary>
01165 /// Save a page to an image file in JPEG format, with the specified quality.
01166 /// </summary>
01167 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01168 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01169 /// pixel of the image.</param>
01170 /// <param name="fileName">The path to the output file.</param>
01171 /// <param name="quality">The quality of the JPEG output file (ranging from 0 to 100).</param>
01172 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01173 /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
01174 /// included.</param>
01175         public void SaveImageAsJPEG(int pageNumber, double zoom, string fileName, int quality, bool
01176             includeAnnotations = true)

```

```

01170         {
01171             if (this.EncryptionState == EncryptionState.Encrypted)
01172             {
01173                 throw new DocumentLockedException("A password is necessary to render the document!");
01174             }
01175
01176             Rectangle region = this.Pages[pageNumber].Bounds;
01177             SaveImageAsJPEG(pageNumber, region, zoom, fileName, quality, includeAnnotations);
01178         }
01179
01180     /// <summary>
01181     /// Write (part of) a page to an image stream in the specified format.
01182     /// </summary>
01183     /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01184     /// <param name="region">The region of the page to render in page units.</param>
01185     /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01186     /// pixel of the image.</param>
01187     /// <param name="pixelFormat">The format of the pixel data.</param>
01188     /// <param name="outputStream">The stream to which the image data will be written.</param>
01189     /// <param name="fileType">The output format of the image.</param>
01190     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01191     /// signatures) are included in the display list that is generated. Otherwise, only the page contents are
01192     /// included.</param>
01193     public void WriteImage(int pageNumber, Rectangle region, double zoom, PixelFormats
01194     pixelFormat, Stream outputStream, RasterOutputFileTypes fileType, bool includeAnnotations = true)
01195     {
01196         if (this.EncryptionState == EncryptionState.Encrypted)
01197         {
01198             throw new DocumentLockedException("A password is necessary to render the document!");
01199         }
01200
01201         if (pixelFormat == PixelFormats.RGBA && fileType == RasterOutputFileTypes.PNM)
01202         {
01203             throw new ArgumentException("Cannot save an image with alpha channel in PNM format!",
01204             nameof(fileType));
01205         }
01206
01207         if (pixelFormat != PixelFormats.RGB && fileType == RasterOutputFileTypes.JPG)
01208         {
01209             throw new ArgumentException("The JPEG format only supports RGB pixel data without an
01210             alpha channel!", nameof(fileType));
01211         }
01212
01213         if (DisplayLists[pageNumber] == null)
01214         {
01215             DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01216             this.Pages[pageNumber], includeAnnotations);
01217         }
01218
01219         if (zoom < 0.000001 || zoom * region.Width <= 0.001 || zoom * region.Height <= 0.001)
01220         {
01221             throw new ArgumentOutOfRangeException(nameof(zoom), zoom, "The zoom factor is too
01222             small!");
01223         }
01224
01225         if (this.ImageXRes != 72 || this.ImageYRes != 72)
01226         {
01227             zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01228             region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 /
01229             this.ImageYRes, region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01230         }
01231
01232         float fzoom = (float)zoom;
01233
01234         IntPtr outputBuffer = IntPtr.Zero;
01235         IntPtr outputData = IntPtr.Zero;
01236         ulong outputDataLength = 0;
01237
01238         ExitCodes result = (ExitCodes)NativeMethods.WriteImage(OwnerContext.NativeContext,
01239             DisplayLists[pageNumber].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, fzoom,
01240             (int)pixelFormat, (int)fileType, 90, ref outputBuffer, ref outputData, ref outputDataLength);
01241
01242         switch (result)
01243         {
01244             case ExitCodes.EXIT_SUCCESS:
01245                 break;
01246             case ExitCodes.ERR_CANNOT_RENDER:
01247                 throw new MuPDFException("Cannot render page", result);
01248             case ExitCodes.ERR_CANNOT_CREATE_BUFFER:
01249                 throw new MuPDFException("Cannot create the output buffer", result);
01250         }

```

```

01244         default:
01245             throw new MuPDFException("Unknown error", result);
01246     }
01247 
01248     byte[] buffer = new byte[1024];
01249 
01250     while (outputDataLength > 0)
01251     {
01252         int bytesToCopy = (int)Math.Min(buffer.Length, (long)outputDataLength);
01253         Marshal.Copy(outputData, buffer, 0, bytesToCopy);
01254         outputData = IntPtr.Add(outputData, bytesToCopy);
01255         outputStream.Write(buffer, 0, bytesToCopy);
01256         outputDataLength -= (ulong)bytesToCopy;
01257     }
01258 
01259     NativeMethods.DisposeBuffer(OwnerContext.NativeContext, outputBuffer);
01260 }
01261 
01262 
01263 /// <summary>
01264 /// Write (part of) a page to an image stream in JPEG format, with the specified quality.
01265 /// </summary>
01266 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01267 /// <param name="region">The region of the page to render in page units.</param>
01268 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01269 /// pixel of the image.</param>
01270 /// <param name="outputStream">The stream to which the image data will be written.</param>
01271 /// <param name="quality">The quality of the JPEG output (ranging from 0 to 100).</param>
01272 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01273 // signatures) are included in the display list that is generated. Otherwise, only the page contents are
01274 // included.</param>
01275 public void WriteImageAsJPEG(int pageNumber, Rectangle region, double zoom, Stream
01276     outputStream, int quality, bool includeAnnotations = true)
01277 {
01278     if (this.EncryptionState == EncryptionState.Encrypted)
01279     {
01280         throw new DocumentLockedException("A password is necessary to render the document!");
01281     }
01282 
01283     if (quality < 0 || quality > 100)
01284     {
01285         throw new ArgumentOutOfRangeException(nameof(quality), quality, "The JPEG quality must
01286 range between 0 and 100 (inclusive)!");
01287     }
01288 
01289     if (DisplayLists[pageNumber] == null)
01290     {
01291         DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01292             this.Pages[pageNumber], includeAnnotations);
01293     }
01294 
01295     if (zoom < 0.000001 || zoom * region.Width <= 0.001 || zoom * region.Height <= 0.001)
01296     {
01297         throw new ArgumentOutOfRangeException(nameof(zoom), zoom, "The zoom factor is too
01298 small!");
01299     }
01300 
01301     if (this.ImageXRes != 72 || this.ImageYRes != 72)
01302     {
01303         zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01304         region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 / this.ImageYRes,
01305             region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01306     }
01307 
01308     float fzoom = (float)zoom;
01309 
01310     IntPtr outputBuffer = IntPtr.Zero;
01311     IntPtr outputData = IntPtr.Zero;
01312     ulong outputDataLength = 0;
01313 
01314     ExitCodes result = (ExitCodes)NativeMethods.WriteImage(OwnerContext.NativeContext,
01315         DisplayLists[pageNumber].NativeDisplayList, region.X0, region.Y0, region.X1, region.Y1, fzoom,
01316         (int)PixelFormats.RGB, (int)RasterOutputFileTypes.JPEG, quality, ref outputBuffer, ref outputData, ref
01317         outputDataLength);
01318 
01319     switch (result)
01320     {
01321         case ExitCodes.EXIT_SUCCESS:
01322             break;
01323         case ExitCodes.ERR_CANNOT_RENDER:
01324             throw new MuPDFException("Cannot render page", result);
01325         case ExitCodes.ERR_CANNOT_CREATE_BUFFER:
01326             throw new MuPDFException("Cannot create the output buffer", result);
01327         default:
01328             throw new MuPDFException("Unknown error", result);
01329     }
01330 }
```

```

01320         byte[] buffer = new byte[1024];
01321
01322         while (outputDataLength > 0)
01323     {
01324             int bytesToCopy = (int) Math.Min(buffer.Length, (long) outputDataLength);
01325             Marshal.Copy(outputData, buffer, 0, bytesToCopy);
01326             outputData = IntPtr.Add(outputData, bytesToCopy);
01327             outputStream.Write(buffer, 0, bytesToCopy);
01328             outputDataLength -= (ulong) bytesToCopy;
01329         }
01330
01331         NativeMethods.DisposeBuffer(OwnerContext.NativeContext, outputBuffer);
01332     }
01333
01334 /// <summary>
01335 /// Write a page to an image stream in the specified format.
01336 /// </summary>
01337 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01338 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01339 /// pixel of the image.</param>
01340 /// <param name="pixelFormat">The format of the pixel data.</param>
01341 /// <param name="outputStream">The stream to which the image data will be written.</param>
01342 /// <param name="fileType">The output format of the image.</param>
01343 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01344 // signatures) are included in the display list that is generated. Otherwise, only the page contents are
01345 // included.</param>
01346         public void WriteImage(int pageNumber, double zoom, PixelFormats pixelFormat, Stream
01347             outputStream, RasterOutputFileTypes fileType, bool includeAnnotations = true)
01348     {
01349         if (this.EncryptionState == EncryptionState.Encrypted)
01350         {
01351             throw new DocumentLockedException("A password is necessary to render the document!");
01352         }
01353
01354         Rectangle region = this.Pages[pageNumber].Bounds;
01355         WriteImage(pageNumber, region, zoom, pixelFormat, outputStream, fileType,
01356             includeAnnotations);
01357     }
01358
01359 /// <summary>
01360 /// Write a page to an image stream in JPEG format, with the specified quality.
01361 /// </summary>
01362 /// <param name="pageNumber">The number of the page to render (starting at 0).</param>
01363 /// <param name="zoom">The scale at which the page will be rendered. This will determine the size in
01364 /// pixel of the image.</param>
01365 /// <param name="outputStream">The stream to which the image data will be written.</param>
01366 /// <param name="quality">The quality of the JPEG output (ranging from 0 to 100).</param>
01367 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01368 // signatures) are included in the display list that is generated. Otherwise, only the page contents are
01369 // included.</param>
01370         public void WriteImageAsJPEG(int pageNumber, double zoom, Stream outputStream, int quality,
01371             bool includeAnnotations = true)
01372     {
01373         if (this.EncryptionState == EncryptionState.Encrypted)
01374         {
01375             throw new DocumentLockedException("A password is necessary to render the document!");
01376         }
01377
01378         Rectangle region = this.Pages[pageNumber].Bounds;
01379         WriteImageAsJPEG(pageNumber, region, zoom, outputStream, quality, includeAnnotations);
01380
01381
01382         if (this.EncryptionState == EncryptionState.Encrypted)
01383         {
01384             throw new DocumentLockedException("A password is necessary to render the document!");
01385         }
01386
01387         if (DisplayLists[pageNumber] == null)
01388         {
01389             DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01390                 this.Pages[pageNumber], includeAnnotations);
01391         }
01392     }
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
02099
02100
02101
02102
02103
02104
02105
02106
02107
02108
02109
02110
02111
02112
02113
02114
02115
02116
02117
02118
02119
02120
02121
02122
02123
02124
02125
02126
02127
02128
02129
02130
02131
02132
02133
02134
02135
02136
02137
02138
02139
02140
02141
02142
02143
02144
02145
02146
02147
02148
02149
02150
02151
02152
02153
02154
02155
02156
02157
02158
02159
02160
02161
02162
02163
02164
02165
02166
02167
02168
02169
02170
02171
02172
02173
02174
02175
02176
02177
02178
02179
02180
02181
02182
02183
02184
02185
02186
02187
02188
02189
02190
02191
02192
02193
02194
02195
02196
02197
02198
02199
02200
02201
02202
02203
02204
02205
02206
02207
02208
02209
02210
02211
02212
02213
02214
02215
02216
02217
02218
02219
02220
02221
02222
02223
02224
02225
02226
02227
02228
02229
02230
02231
02232
02233
02234
02235
02236
02237
02238
02239
02240
02241
02242
02243
02244
02245
02246
02247
02248
02249
02250
02251
02252
02253
02254
02255
02256
02257
02258
02259
02260
02261
02262
02263
02264
02265
02266
02267
02268
02269
02270
02271
02272
02273
02274
02275
02276
02277
02278
02279
02280
02281
02282
02283
02284
02285
02286
02287
02288
02289
02290
02291
02292
02293
02294
02295
02296
02297
02298
02299
02300
02301
02302
02303
02304
02305
02306
02307
02308
02309
02310
02311
02312
02313
02314
02315
02316
02317
02318
02319
02320
02321
02322
02323
02324
02325
02326
02327
02328
02329
02330
02331
02332
02333
02334
02335
02336
02337
02338
02339
02340
02341
02342
02343
02344
02345
02346
02347
02348
02349
02350
02351
02352
02353
02354
02355
02356
02357
02358
02359
02360
02361
02362
02363
02364
02365
02366
02367
02368
02369
02370
02371
02372
02373
02374
02375
02376
02377
02378
02379
02380
02381
02382
02383
02384
02385
02386
02387
02388
02389
02390
02391
02392
02393
02394
02395
02396
02397
02398
02399
02400
02401
02402
02403
02404
02405
02406
02407
02408
02409
02410
02411
02412
02413
02414
02415
02416
02417
02418
02419
02420
02421
02422
02423
02424
02425
02426
02427
02428
02429
02430
02431
02432
02433
02434
02435
02436
02437
02438
02439
02440
02441
02442
02443
02444
02445
02446
02447
02448
02449
02450
02451
02452
02453
02454
02455
02456
02457
02458
02459
02460
02461
02462
02463
02464
02465
02466
02467
02468
02469
02470
02471
02472
02473
02474
02475
02476
02477
02478
02479
02480
02481
02482
02483
02484
02485
02486
02487
02488
02489
02490
02491
02492
02493
02494
02495
02496
02497
02498
02499
02500
02501
02502
02503
02504
02505
02506
02507
02508
02509
02510
02511
02512
02513
02514
02515
02516
02517
02518
02519
02520
02521
02522
02523
02524
02525
02526
02527
02528
02529
02530
02531
02532
02533
02534
02535
02536
02537
02538
02539
02540
02541
02542
02543
02544
02545
02546
02547
02548
02549
02550
02551
02552
02553
02554
02555
02556
02557
02558
02559
02560
02561
02562
02563
02564
02565
02566
02567
02568
02569
02570
02571
02572
02573
02574
02575
02576
02577
02578
02579
02580
02581
02582
02583
02584
02585
02586
02587
02588
02589
02590
02591
02592
02593
02594
02595
02596
02597
02598
02599
02600
02601
02602
02603
02604
02605
02606
02607
02608
02609
02610
02611
02612
02613
02614
02615
02616
02617
02618
02619
02620
02621
02622
02623
02624
02625
02626
02627
02628
02629
02630
02631
02632
02633
02634
02635
02636
02637
02638
02639
02640
02641
02642
02643
02644
02645
02646
02647
02648
02649
02650
02651
02652
02653
02654
02655
02656
02657
02658
02659
02660
02661
02662
02663
02664
02665
02666
02667
02668
02669
02670
02671
02672
02673
02674
02675
02676
02677
02678
02679
02680
02681
02682
02683
02684
02685
02686
02687
02688
02689
02690
02691
02692
02693
02694
02695
02696
02697
02698
02699
02700
02701
02702
02703
02704
02705
02706
02707
02708
02709
02710
02711
02712
02713
02714
02715
02716
02717
02718
02719
02720
02721
02722
02723
02724
02725
02726
02727
02728
02729
02730
02731
02732
02733
02734
02735
02736
02737
02738
02739
02740
02741
02742
02743
02744
02745
02746
02747
02748
02749
02750
02751
02752
02753
02754
02755
02756
02757
02758
02759
02760
02761
02762
02763
02764
02765
02766
02767
02768
02769
02770
02771
02772
02773
02774
02775
02776
02777
02778
02779
02780
02781
02782
02783
02784
02785
02786
02787
02788
02789
02790
02791
02792
02793
02794
02795
02796
02797
02798
02799
02800
02801
02802
02803
02804
02805
02806
02807
02808
02809
02810
02811
02812
02813
02814
02815
02816
02817
02818
02819
02820
02821
02822
02823
02824
02825
02826
02827
02828
02829
02830
02831
02832
02833
02834
02835
02836
02837
02838
02839
02840
02841
02842
02843
02844
02845
02846
02847
02848
02849
02850
02851
02852
02853
02854
02855
02856
02857
02858
02859
02860
02861
02862
02863
02864
02865
02866
02867
02868
02869
02870
02871
02872
02873
02874
02875
02876
02877
02878
02879
02880
02881
02882
02883
02884
02885
02886
02887
02888
02889
02890
02891
02892
02893
02894
02895
02896
02897
02898
02899
02900
02901
02902
02903
02904
02905
02906
02907
02908
02909
02910
02911
02912
02913
02914
02915
02916
02917
02918
02919
02920
02921
02922
02923
02924
02925
02926
02927
02928
02929
02930
02931
02932
02933
02934
02935
02936
02937
02938
02939
02940
02941
02942
02943
02944
02945
02946
02947
02948
02949
02950
02951
02952
02953
02954
02955
02956
02957
02958
02959
02960
02961
02962
02963
02964
02965
02966
02967
02968
02969
02970
02971
02972
02973
02974
02975
02976
02977
02978
02979
02980
02981
02982
02983
02984
02985
02986
02987
02988
02989
02990
02991
02992
02993
02994
02995
02996
02997
02998
02999
03000
03001
03002
03003
03004
03005
03006
03007
03008
03009
03010
03011
03012
03013
03014
03015
03016
03017
03018
03019
03020
03021
03022
03023
03024
03025
03026
03027
03028
03029
03030
03031
03032
03033
03034
03035
03036
03037
03
```

```
01392         return new MuPDFStructuredTextPage(this.OwnerContext, this.DisplayLists[pageNumber], null,
01393             1, new Rectangle(), flags);
01394     }
01395 /// <summary>
01396 /// Creates a new <see cref="MuPDFStructuredTextPage"/> from the specified page, using optical
01397     character recognition (OCR) to determine what text is written on the image. This contains information
01398     about the text layout that can be used for highlighting and searching.
01399 /// </summary>
01400 /// <param name="pageNumber">The number of the page (starting at 0)</param>
01401 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
01402     null, no OCR is performed.</param>
01403 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01404     signatures) are included. Otherwise, only the page contents are included.</param>
01405 /// <param name="flags">Flags for the structured text extraction process.</param>
01406 /// <param name="cancellationToken">A <see cref="CancellationToken"/> used to cancel the operation.
01407     Providing a value other than the default is not supported on Windows x86 and will throw a runtime
01408     exception.</param>
01409 /// <param name="progress">An <see cref="IProgress<OCRProgressInfo>"/> used to report progress.
01410     Providing a value other than null is not supported on Windows x86 and will throw a runtime
01411     exception.</param>
01412 /// <returns>A <see cref="MuPDFStructuredTextPage"/> containing a structured text representation of
01413     the page.</returns>
01414     public MuPDFStructuredTextPage GetStructuredTextPage(int pageNumber, TesseractLanguage
01415     ocrLanguage, bool includeAnnotations = true, StructuredTextFlags flags = StructuredTextFlags.None,
01416     CancellationToken cancellationToken = default, IProgress<OCRProgressInfo> progress = null)
01417     {
01418         if (this.EncryptionState == EncryptionState.Encrypted)
01419         {
01420             throw new DocumentLockedException("A password is necessary to render the document!");
01421         }
01422         if (DisplayLists[pageNumber] == null)
01423         {
01424             DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01425                 this.Pages[pageNumber], includeAnnotations);
01426         }
01427         double zoom = 1;
01428         Rectangle region = this.Pages[pageNumber].Bounds;
01429         if (this.ImageXRes != 72 || this.ImageYRes != 72)
01430         {
01431             zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01432             region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 / this.ImageYRes,
01433                 region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01434         }
01435         return new MuPDFStructuredTextPage(this.OwnerContext, this.DisplayLists[pageNumber],
01436             ocrLanguage, zoom, region, flags, cancellationToken, progress);
01437     }
01438 /// <summary>
01439 /// Creates a new <see cref="MuPDFStructuredTextPage"/> from the specified page, using optical
01440     character recognition (OCR) to determine what text is written on the image. This contains information
01441     about the text layout that can be used for highlighting and searching. The OCR step is run
01442     asynchronously, e.g. to avoid blocking the UI thread.
01443 /// </summary>
01444 /// <param name="pageNumber">The number of the page (starting at 0)</param>
01445 /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
01446     null, no OCR is performed.</param>
01447 /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01448     signatures) are included. Otherwise, only the page contents are included.</param>
01449 /// <param name="flags">Flags for the structured text extraction process.</param>
01450 /// <param name="cancellationToken">A <see cref="CancellationToken"/> used to cancel the operation.
01451     Providing a value other than the default is not supported on Windows x86 and will throw a runtime
01452     exception.</param>
01453 /// <param name="progress">An <see cref="IProgress<OCRProgressInfo>"/> used to report progress.
01454     Providing a value other than null is not supported on Windows x86 and will throw a runtime
01455     exception.</param>
01456 /// <returns>A <see cref="MuPDFStructuredTextPage"/> containing a structured text representation of
01457     the page.</returns>
01458     public async Task<MuPDFStructuredTextPage> GetStructuredTextPageAsync(int pageNumber,
01459         TesseractLanguage ocrLanguage, bool includeAnnotations = true, StructuredTextFlags flags =
01460             StructuredTextFlags.None, CancellationToken cancellationToken = default, IProgress<OCRProgressInfo>
01461             progress = null)
01462     {
01463         if (this.EncryptionState == EncryptionState.Encrypted)
01464         {
01465             throw new DocumentLockedException("A password is necessary to render the document!");
01466         }
01467         if (DisplayLists[pageNumber] == null)
01468         {
01469             DisplayLists[pageNumber] = new MuPDFDisplayList(this.OwnerContext,
01470                 this.Pages[pageNumber], includeAnnotations);
01471         }
01472     }
```

```

01450             double zoom = 1;
01451             Rectangle region = this.Pages[pageNumber].Bounds;
01452
01453             if (this.ImageXRes != 72 || this.ImageYRes != 72)
01454             {
01455                 zoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
01456                 region = new Rectangle(region.X0 * 72 / this.ImageXRes, region.Y0 * 72 / this.ImageYRes,
01457                                         region.X1 * 72 / this.ImageXRes, region.Y1 * 72 / this.ImageYRes);
01458             }
01459
01460             return await Task.Run(() => new MuPDFStructuredTextPage(this.OwnerContext,
01461                                         this.DisplayLists[pageNumber], ocrLanguage, zoom, region, flags, cancellationToken, progress));
01462         }
01463     /// <summary>
01464     /// Extracts all the text from the document and returns it as a <see cref="string"/>. The reading
01465     /// order is taken from the order the text is drawn in the source file, so may not be accurate.
01466     /// </summary>
01467     /// <param name="separator">The character(s) used to separate the text lines obtained from the
01468     /// document. If this is <see langword="null" />, <see cref="Environment.NewLine"/> is used as a default
01469     /// separator.</param>
01470     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01471     /// signatures) are included. Otherwise, only the page contents are included.</param>
01472     /// <returns>A <see cref="string"/> containing all the text in the document. Characters are converted
01473     /// from the UTF-8 representation used in the document to equivalent UTF-16 <see
01474     /// cref="string"/>s.</returns>
01475     public string ExtractText(string separator = null, bool includeAnnotations = true)
01476     {
01477         if (this.EncryptionState == EncryptionState.Encrypted)
01478         {
01479             throw new DocumentLockedException("A password is necessary to render the document!");
01480         }
01481
01482         separator = separator ?? Environment.NewLine;
01483
01484         var text = new StringBuilder();
01485         bool started = false;
01486
01487         for (int i = 0; i < this.Pages.Count; i++)
01488         {
01489             using (MuPDFStructuredTextPage structuredTextPage = this.GetStructuredTextPage(i,
01490                                         includeAnnotations, StructuredTextFlags.None))
01491             {
01492                 foreach (MuPDFStructuredTextBlock textBlock in
01493                         structuredTextPage.StructuredTextBlocks)
01494                 {
01495                     var numLines = textBlock.Count;
01496                     for (var j = 0; j < numLines; j++)
01497                     {
01498                         if (!string.IsNullOrWhiteSpace(textBlock[j].Text))
01499                         {
01500                             if (started)
01501                             {
01502                                 text.Append(separator);
01503                             }
01504                         }
01505                     }
01506                 }
01507
01508             return text.ToString();
01509         }
01510     }
01511     /// <summary>
01512     /// Extracts all the text from the document and returns it as a <see cref="string"/>, using optical
01513     /// character recognition (OCR) to determine what text is written on the image.
01514     /// </summary>
01515     /// <param name="separator">The character(s) used to separate the text lines obtained from the
01516     /// document. If this is <see langword="null" />, <see cref="Environment.NewLine"/> is used as a default
01517     /// separator.</param>
01518     /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
01519     /// null, no OCR is performed.</param>
01520     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01521     /// signatures) are included. Otherwise, only the page contents are included.</param>
01522     /// <returns>A <see cref="string"/> containing all the text in the document. Characters are converted
01523     /// from the UTF-8 representation used in the document to equivalent UTF-16 <see
01524     /// cref="string"/>s.</returns>
01525     public string ExtractText(TesseractLanguage ocrLanguage, string separator = null, bool
01526                               includeAnnotations = true)

```

```
01519         {
01520             if (this.EncryptionState == EncryptionState.Encrypted)
01521             {
01522                 throw new DocumentLockedException("A password is necessary to render the document!");
01523             }
01524
01525             separator = separator ?? Environment.NewLine;
01526
01527             var text = new StringBuilder();
01528             bool started = false;
01529
01530             for (int i = 0; i < this.Pages.Count; i++)
01531             {
01532                 using (MuPDFStructuredTextPage structuredTextPage = this.GetStructuredTextPage(i,
01533                     ocrLanguage, includeAnnotations, StructuredTextFlags.None))
01534                 {
01535                     foreach (MuPDFStructuredTextBlock textBlock in
01536                     structuredTextPage.StructuredTextBlocks)
01537                     {
01538                         var numLines = textBlock.Count;
01539                         for (var j = 0; j < numLines; j++)
01540                         {
01541                             if (!string.IsNullOrWhiteSpace(textBlock[j].Text))
01542                             {
01543                                 if (started)
01544                                 {
01545                                     text.Append(separator);
01546                                 }
01547                                 else
01548                                 {
01549                                     started = true;
01550                                 }
01551                             }
01552                         }
01553                     }
01554                 }
01555             }
01556
01557             return text.ToString();
01558         }
01559
01560     /// <summary>
01561     /// Extracts all the text from the document and returns it as a <see cref="string"/>, using optical
01562     /// character recognition (OCR) to determine what text is written on the image. The OCR step is run
01563     /// asynchronously, e.g. to avoid blocking the UI thread.
01564     /// </summary>
01565     /// <param name="separator">The character(s) used to separate the text lines obtained from the
01566     /// document. If this is <see langword="null" />, <see cref="Environment.NewLine"/> is used as a default
01567     /// separator.</param>
01568     /// <param name="ocrLanguage">The language to use for optical character recognition (OCR). If this is
01569     /// null, no OCR is performed.</param>
01570     /// <param name="includeAnnotations">If this is <see langword="true" />, annotations (e.g.
01571     /// signatures) are included. Otherwise, only the page contents are included.</param>
01572     /// <param name="cancellationToken">A <see cref="CancellationToken"/> used to cancel the
01573     /// operation.</param>
01574     /// <param name="progress">An <see cref="IProgress<OCRProgressInfo>"/> used to report
01575     /// progress.</param>
01576     /// <returns>A <see cref="string"/> containing all the text in the document. Characters are converted
01577     /// from the UTF-8 representation used in the document to equivalent UTF-16 <see
01578     /// cref="string"/>s.</returns>
01579     public async Task<string> ExtractTextAsync(TesseractLanguage ocrLanguage, string separator =
01580         null, bool includeAnnotations = true, CancellationToken cancellationToken = default,
01581         IProgress<OCRProgressInfo> progress = null)
01582     {
01583         if (this.EncryptionState == EncryptionState.Encrypted)
01584         {
01585             throw new DocumentLockedException("A password is necessary to render the document!");
01586         }
01587
01588         separator = separator ?? Environment.NewLine;
01589
01590         var text = new StringBuilder();
01591         bool started = false;
01592
01593         for (int i = 0; i < this.Pages.Count; i++)
01594         {
01595             using (MuPDFStructuredTextPage structuredTextPage = await
01596                 this.GetStructuredTextPageAsync(i, ocrLanguage, includeAnnotations, StructuredTextFlags.None,
01597                 cancellationToken, progress))
01598             {
01599                 foreach (MuPDFStructuredTextBlock textBlock in
01600                     structuredTextPage.StructuredTextBlocks)
01601                 {
01602                     var numLines = textBlock.Count;
01603                     for (var j = 0; j < numLines; j++)
```

```

01589             {
01590                 if (!string.IsNullOrWhiteSpace(textBlock[j].Text))
01591                 {
01592                     if (started)
01593                     {
01594                         text.Append(separator);
01595                     }
01596                     else
01597                     {
01598                         started = true;
01599                     }
01600                 text.Append(textBlock[j].Text);
01601             }
01602         }
01603     }
01604 }
01605 }
01606 }
01607
01608     return text.ToString();
01609 }
01610
01611 /// <summary>
01612 /// Attempts to unlock the document with the supplied password.
01613 /// </summary>
01614 /// <param name="password">The user or owner password to use to unlock the document.</param>
01615 /// <returns><see langword="true"/>If the document was successfully unlocked (or if it was never
locked to begin with), <see langword="false"/> if the password was incorrect and the document is still
locked.</returns>
01616 /// <remarks>This method can be used both to unlock an encrypted document and to check whether the
supplied owner password is correct.</remarks>
01617     public bool TryUnlock(string password)
01618     {
01619         return TryUnlock(password, out _);
01620     }
01621
01622 /// <summary>
01623 /// Attempts to unlock the document with the supplied password.
01624 /// </summary>
01625 /// <param name="password">The user or owner password to use to unlock the document.</param>
01626 /// <param name="passwordType">If the method returns true, this can be used to determine whether the
supplied password was the user password or the owner password. If the method returns <see
langword="false"/>,
01627 /// this can be used to determine whether a user password and/or an owner password are
required.</param>
01628 /// <returns><see langword="true"/>If the document was successfully unlocked (or if it was never
locked to begin with), <see langword="false"/> if the password was incorrect and the document is still
locked.</returns>
01629 /// <remarks>This method can be used both to unlock an encrypted document and to check whether the
supplied owner password is correct.</remarks>
01630     public bool TryUnlock(string password, out PasswordTypes passwordType)
01631     {
01632         int result = NativeMethods.UnlockWithPassword(this.OwnerContext.NativeContext,
this.NativeDocument, password);
01633
01634         int pt;
01635
01636         switch (result)
01637         {
01638             case 0:
01639                 pt = 0;
01640
01641                 if (this.EncryptionState == EncryptionState.Encrypted)
01642                 {
01643                     pt |= 1;
01644                 }
01645
01646                 if (this.RestrictionState == RestrictionState.Restricted)
01647                 {
01648                     pt |= 2;
01649                 }
01650
01651                 passwordType = (PasswordTypes)pt;
01652                 return !(this.EncryptionState == EncryptionState.Encrypted ||
this.RestrictionState == RestrictionState.Restricted);
01653             case 1:
01654                 passwordType = PasswordTypes.None;
01655                 return true;
01656             case 2:
01657                 if (this.EncryptionState == EncryptionState.Encrypted)
01658                 {
01659                     this.EncryptionState = EncryptionState.Unlocked;
01660                 }
01661                 passwordType = PasswordTypes.User;
01662                 return true;
01663             case 4:
01664                 if (this.RestrictionState == RestrictionState.Restricted)

```

```
01665             {
01666                 this.RestrictionState = RestrictionState.Unlocked;
01667             }
01668             passwordType = PasswordTypes.Owner;
01669             return true;
01670         case 6:
01671             pt = 0;
01672
01673             if (this.EncryptionState == EncryptionState.Encrypted)
01674             {
01675                 this.EncryptionState = EncryptionState.Unlocked;
01676                 pt |= 1;
01677             }
01678
01679             if (this.RestrictionState == RestrictionState.Restricted)
01680             {
01681                 this.RestrictionState = RestrictionState.Unlocked;
01682                 pt |= 2;
01683             }
01684             passwordType = (PasswordTypes)pt;
01685             return true;
01686         default:
01687             throw new ArgumentOutOfRangeException("Unexpected return value when unlocking the
document: " + result.ToString());
01688         }
01689     }
01690
01691     private bool disposedValue;
01692
01693 ///<inheritdoc/>
01694     protected virtual void Dispose(bool disposing)
01695     {
01696         if (!disposedValue)
01697         {
01698             if (disposing)
01699             {
01700                 Pages.Dispose();
01701                 foreach (MuPDFDisplayList list in DisplayLists)
01702                 {
01703                     list?.Dispose();
01704                 }
01705                 DataHandle?.Free();
01706                 DataHolder?.Dispose();
01707             }
01708
01709             if (OwnerContext.disposedValue)
01710             {
01711                 throw new LifetimeManagementException<MuPDFDocument, MuPDFContext>(this,
OwnerContext, this.NativeDocument, OwnerContext.NativeContext);
01712             }
01713
01714             if (NativePDFDocument != IntPtr.Zero)
01715             {
01716                 NativeMethods.DisposeDocument(OwnerContext.NativeContext, NativePDFDocument);
01717             }
01718
01719             NativeMethods.DisposeDocument(OwnerContext.NativeContext, NativeDocument);
01720
01721             if (NativeStream != IntPtr.Zero)
01722             {
01723                 NativeMethods.DisposeStream(OwnerContext.NativeContext, NativeStream);
01724             }
01725
01726             disposedValue = true;
01727         }
01728     }
01729 }
01730
01731 ///<inheritdoc/>
01732     ~MuPDFDocument()
01733     {
01734         Dispose(disposing: false);
01735     }
01736
01737 ///<inheritdoc/>
01738     public void Dispose()
01739     {
01740         Dispose(disposing: true);
01741         GC.SuppressFinalize(this);
01742     }
01743 }
01744 }
```

8.10 MuPDFFont.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2024 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections.Generic;
00020 using System.Text;
00021
00022
00023 namespace MuPDFCore
00024 {
00025 /// <summary>
00026 /// Represents a font.
00027 /// </summary>
00028 public class MuPDFFont : IDisposable
00029 {
00030     internal bool disposedValue;
00031
00032 /// <summary>
00033 /// Returns whether the font is bold or not.
00034 /// </summary>
00035     public bool IsBold { get; }
00036
00037 /// <summary>
00038 /// Returns whether the font is italic or not.
00039 /// </summary>
00040     public bool IsItalic { get; }
00041
00042 /// <summary>
00043 /// Returns whether the font is serif or not.
00044 /// </summary>
00045     public bool IsSerif { get; }
00046
00047 /// <summary>
00048 /// Returns whether the font is monospaced or not.
00049 /// </summary>
00050     public bool IsMonospaced { get; }
00051
00052 /// <summary>
00053 /// The name of the font.
00054 /// </summary>
00055     public string Name { get; }
00056
00057     private MuPDFContext OwnerContext { get; }
00058     private IntPtr NativePointer { get; }
00059
00060     internal unsafe MuPDFFont(MuPDFContext context, IntPtr nativePointer)
00061     {
00062         int nameLength = 0;
00063         int bold = -1;
00064         int italic = -1;
00065         int serif = -1;
00066         int monospaced = -1;
00067
00068         ExitCodes result = (ExitCodes)NativeMethods.GetFontMetadata(context.NativeContext,
00069                         nativePointer, ref nameLength, ref bold, ref italic, ref serif, ref monospaced);
00070
00071         switch (result)
00072         {
00073             case ExitCodes.EXIT_SUCCESS:
00074                 break;
00075             case ExitCodes.ERR_FONT_METADATA:
00076                 throw new MuPDFException("An error occurred while retrieving font metadata!",
00077                                         result);
00078             default:
00079                 throw new MuPDFException("Unknown error!", result);
00080         }
00081
00082         byte[] fontName = new byte[nameLength];
00083
00084         fixed (byte* fontNamePtr = fontName)
00085         {

```

```
00084         result = (ExitCodes)NativeMethods.GetFontName(context.NativeContext, nativePointer,
00085             nameLength, (IntPtr)fontNamePtr);
00086     }
00087     switch (result)
00088     {
00089         case ExitCodes.EXIT_SUCCESS:
00090             break;
00091         case ExitCodes.ERR_FONT_METADATA:
00092             throw new MuPDFException("An error occurred while retrieving font metadata!", result);
00093         default:
00094             throw new MuPDFException("Unknown error!", result);
00095     }
00096     this.IsBold = bold != 0;
00097     this.IsItalic = italic != 0;
00098     this.IsSerif = serif != 0;
00099     this.IsMonospaced = monospaced != 0;
00100     this.Name = Encoding.UTF8.GetString(fontName);
00101     this.OwnerContext = context;
00102     this.NativePointer = nativePointer;
00103 }
00104
00105
00106
00107 /// <summary>
00108 /// Get a pointer to the FreeType FT_Face object for this font. You will need native bindings to the
00109 /// FreeType library to use this.
00110 /// </summary>
00111 /// <returns>A pointer to the FreeType FT_Face object for this font, or <see cref="IntPtr.Zero"/> if
00112 /// this is a Type3 font.</returns>
00113 /// <exception cref="MuPDFException">Thrown if an error occurs while accessing the FreeType handle for
00114 /// the font.</exception>
00115     public IntPtr GetFreeTypeHandle()
00116     {
00117         if (disposedValue)
00118         {
00119             throw new ObjectDisposedException("MuPDFFont", "The MuPDFFont object has already been
00120 disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00121         }
00122         IntPtr tbr = IntPtr.Zero;
00123         ExitCodes result = (ExitCodes)NativeMethods.GetFTHandle(this.OwnerContext.NativeContext,
00124             this.NativePointer, ref tbr);
00125         switch (result)
00126         {
00127             case ExitCodes.EXIT_SUCCESS:
00128                 break;
00129             case ExitCodes.ERR_FONT_METADATA:
00130                 throw new MuPDFException("Cannot get the font handle!", result);
00131             default:
00132                 throw new MuPDFException("Unknown error", result);
00133         }
00134     }
00135
00136 /// <summary>
00137 /// Get a pointer to the Type3 procs for the font. You will need some more specialised MuPDF bindings
00138 /// to do anything with it.
00139 /// </summary>
00140 /// <returns>A pointer to the Type3 procs for the font, or <see cref="IntPtr.Zero"/> if this is not a
00141 /// Type3 font.</returns>
00142 /// <exception cref="MuPDFException">Thrown if an error occurs while accessing the Type3 font
00143 /// procs.</exception>
00144     public IntPtr GetType3Handle()
00145     {
00146         if (disposedValue)
00147         {
00148             throw new ObjectDisposedException("MuPDFFont", "The MuPDFFont object has already been
00149 disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00150         }
00151         IntPtr tbr = IntPtr.Zero;
00152         ExitCodes result = (ExitCodes)NativeMethods.GetT3Procs(this.OwnerContext.NativeContext,
00153             this.NativePointer, ref tbr);
00154         switch (result)
00155         {
00156             case ExitCodes.EXIT_SUCCESS:
00157                 break;
00158             case ExitCodes.ERR_FONT_METADATA:
00159                 throw new MuPDFException("Cannot get the font handle!", result);
00160             default:
```

```

00159         }
00160     }
00161
00162     return tbr;
00163 }
00164
00165 /// <inheritdoc/>
00166     protected virtual void Dispose(bool disposing)
00167     {
00168         if (!disposedValue)
00169         {
00170             if (OwnerContext.disposedValue)
00171             {
00172                 throw new LifetimeManagementException<MuPDFFont, MuPDFContext>(this, OwnerContext,
00173                     this.NativePointer, OwnerContext.NativeContext);
00174             }
00175             lock (OwnerContext.FontCacheLock)
00176             {
00177                 (MuPDFFont font, int referenceCount) item;
00178                 if (OwnerContext.FontCache.TryGetValue(this.NativePointer, out item))
00179                 {
00180                     if (item.referenceCount <= 1)
00181                     {
00182                         NativeMethods.DisposeFont(this.OwnerContext.NativeContext,
00183                             this.NativePointer);
00184                         OwnerContext.FontCache.Remove(this.NativePointer);
00185                         disposedValue = true;
00186                     }
00187                     else
00188                     {
00189                         item = (item.font, item.referenceCount - 1);
00190                         OwnerContext.FontCache[this.NativePointer] = item;
00191                     }
00192                 }
00193             }
00194         }
00195
00196 /// <summary>
00197 /// Dispose the font object.
00198 /// </summary>
00199     ~MuPDFFont()
00200     {
00201         Dispose(disposing: false);
00202     }
00203
00204 /// <inheritdoc/>
00205     public void Dispose()
00206     {
00207         Dispose(disposing: true);
00208         GC.SuppressFinalize(this);
00209     }
00210 }
00211 }

```

8.11 MuPDFImage.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2024 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Runtime.InteropServices;
00020 using System.IO;
00021 using MuPDFCore.StructuredText;
00022
00023 namespace MuPDFCore
00024 {
00025 /// <summary>

```

```
00026 /// Represents an image embedded within a document.
00027 /// </summary>
00028     public class MuPDFImage : IDisposable
00029     {
00030         private bool disposedValue;
00031
00032     /// <summary>
00033     /// Describes the orientation of the image (as encoded within the image file).
00034     /// </summary>
00035         public enum ImageOrientation
00036         {
00037     /// <summary>
00038     /// Undefined.
00039     /// </summary>
00040         Undefined = 0,
00041
00042     /// <summary>
00043     /// 0 degree counter clockwise rotation.
00044     /// </summary>
00045         CCW_0_Degrees = 1,
00046
00047     /// <summary>
00048     /// 90 degree counter clockwise rotation.
00049     /// </summary>
00050         CCW_90_Degrees = 2,
00051
00052     /// <summary>
00053     /// 180 degree counter clockwise rotation.
00054     /// </summary>
00055         CCW_180_Degrees = 3,
00056
00057     /// <summary>
00058     /// 270 degree counter clockwise rotation.
00059     /// </summary>
00060         CCW_270_Degrees = 4,
00061
00062     /// <summary>
00063     /// Flip on X.
00064     /// </summary>
00065         FlipX = 5,
00066
00067     /// <summary>
00068     /// Flip on X, then rotate counter clockwise by 90 degrees.
00069     /// </summary>
00070         FlipX_CCW_90_Degrees = 6,
00071
00072     /// <summary>
00073     /// Flip on X, then rotate counter clockwise by 180 degrees.
00074     /// </summary>
00075         FlipX_CCW_180_Degrees = 7,
00076
00077     /// <summary>
00078     /// Flip on X, then rotate counter clockwise by 270 degrees.
00079     /// </summary>
00080         FlipX_CCW_270_Degrees = 8
00081     }
00082
00083     /// <summary>
00084     /// Width of the image in pixels.
00085     /// </summary>
00086         public int Width { get; }
00087
00088     /// <summary>
00089     /// Height of the image in pixels.
00090     /// </summary>
00091         public int Height { get; }
00092
00093     /// <summary>
00094     /// Horizontal resolution of the image.
00095     /// </summary>
00096         public int XRes { get; }
00097
00098     /// <summary>
00099     /// Vertical resolution of the image.
00100    /// </summary>
00101         public int YRes { get; }
00102
00103     /// <summary>
00104     /// Orientation of the image (as encoded within the image file).
00105     /// </summary>
00106         public ImageOrientation Orientation { get; }
00107
00108     /// <summary>
00109     /// The colour space in which the image is defined.
00110     /// </summary>
00111         public MuPDFColorSpace ColorSpace { get; }
00112
```

```

0013 /// <summary>
0014 /// The <see cref="MuPDFImageStructuredTextBlock"/> from which this image was obtained.
0015 /// </summary>
0016     public MuPDFImageStructuredTextBlock ParentBlock { get; }
0017
0018     private IntPtr NativePointer { get; }
0019     private MuPDFContext OwnerContext { get; }
0020
0021     internal MuPDFImage(IntPtr nativePointer, MuPDFContext context, MuPDFImageStructuredTextBlock
parent)
0022     {
0023         this.NativePointer = nativePointer;
0024         this.OwnerContext = context;
0025
0026         int w = 0;
0027         int h = 0;
0028         int xres = 0;
0029         int yres = 0;
0030         byte orientation = 0;
0031         IntPtr colorspace = IntPtr.Zero;
0032
0033         ExitCodes result = (ExitCodes)NativeMethods.GetImageMetadata(context.NativeContext,
nativePointer, ref w, ref h, ref xres, ref yres, ref orientation, ref colorspace);
0034
0035         switch (result)
0036         {
0037             case ExitCodes.EXIT_SUCCESS:
0038                 break;
0039             case ExitCodes.ERR_IMAGE_METADATA:
0040                 throw new MuPDFException("Error while gathering image metadata.", result);
0041             default:
0042                 throw new MuPDFException("Unknown error", result);
0043         }
0044
0045         this.Width = w;
0046         this.Height = h;
0047         this.XRes = xres;
0048         this.YRes = yres;
0049
0050         this.Orientation = (ImageOrientation)orientation;
0051
0052         this.ColorSpace = MuPDFColorSpace.Create(OwnerContext.NativeContext, colorspace);
0053
0054         NativeMethods.DisposeColorSpace(OwnerContext.NativeContext, colorspace);
0055
0056         this.ParentBlock = parent;
0057     }
0058
0059 /// <summary>
0060 /// Save the image to a file.
0061 /// </summary>
0062 /// <param name="fileName">The name of the output file.</param>
0063 /// <param name="fileType">The output file format.</param>
0064 /// <param name="convertToRGB">If this is <see langword="true"/>, the image is converted to the RGB
colour space before being saved. If this is <see langword="false"/>, the image is saved in the same
colour space as it was encoded in the document. If this is <see langword="null"/> (the default), the
image is converted to RGB only if the target colour space does not support the colour space of the
image.</param>
0065 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image or saving
it.</exception>
0066 /// <exception cref="ArgumentException">Thrown if attempting to export an image in a format that does
not support the colour space of the image.</exception>
0067     public void Save(string fileName, RasterOutputFileTypes fileType, bool? convertToRGB = null)
0068     {
0069         if (this.disposedValue)
0070         {
0071             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
0072         }
0073
0074         if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.CMYK && (fileType ==
RasterOutputFileTypes.PNG || fileType == RasterOutputFileTypes.PNM))
0075         {
0076             if (convertToRGB == false)
0077             {
0078                 throw new ArgumentException("Images in CMYK colour space cannot be exported to PNG
or PNM files!");
0079             }
0080             else if (convertToRGB == null)
0081             {
0082                 convertToRGB = true;
0083             }
0084         }
0085         else if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.Separation && fileType !=
RasterOutputFileTypes.PSD)
0086         {
0087             if (convertToRGB == false)
0088             {
0089                 convertToRGB = true;
0090             }
0091         }
0092     }
0093 }
```

```

00188             {
00189                 throw new ArgumentException("Images in Separation colour space can only be
00190                 exported in PSD format!");
00191             }
00192             else if (convertToRGB == null)
00193             {
00194                 convertToRGB = true;
00195             }
00196         }
00197         ExitCodes result = (ExitCodes)NativeMethods.SaveRasterImage(OwnerContext.NativeContext,
00198             this.NativePointer, fileName, (int)fileType, 90, convertToRGB == true ? 1 : 0);
00199         switch (result)
00200         {
00201             case ExitCodes.EXIT_SUCCESS:
00202                 break;
00203             case ExitCodes.ERR_CANNOT_RENDER:
00204                 throw new MuPDFException("An error occurred while rendering the image.", result);
00205             case ExitCodes.ERR_CANNOT_SAVE:
00206                 throw new MuPDFException("An error occurred while saving the image.", result);
00207             default:
00208                 throw new MuPDFException("Unknown error", result);
00209         }
00210     }
00211 }
00212 /// <summary>
00213 /// Save the image to a JPEG file.
00214 /// </summary>
00215 /// <param name="fileName">The name of the output file.</param>
00216 /// <param name="quality">The quality of the JPEG image (from 0 to 100).</param>
00217 /// <param name="convertToRGB">If this is <see langword="true"/>, the image is converted to the RGB
00218 colour space before being saved. If this is <see langword="false"/>, the image is saved in the same
00219 colour space as it was encoded in the document. If this is <see langword="null"/> (the default), the
00220 image is converted to RGB only if the target colour space does not support the colour space of the
00221 image.</param>
00222 /// <exception cref="ArgumentOutOfRangeException">Thrown if the quality value is < 0 or >
00223 0</exception>
00224 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image or saving
00225 it.</exception>
00226     public void SaveAsJPEG(string fileName, int quality, bool? convertToRGB = null)
00227     {
00228         if (this.disposedValue)
00229         {
00230             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
00231 been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00232         }
00233         if (quality < 0 || quality > 100)
00234         {
00235             throw new ArgumentOutOfRangeException(nameof(quality), quality, "The JPEG quality must
00236 range between 0 and 100 (inclusive)!");
00237         }
00238         if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.Separation)
00239         {
00240             if (convertToRGB == false)
00241             {
00242                 throw new ArgumentException("Images in Separation colour space can only be
00243                 exported in PSD format!");
00244             }
00245             else if (convertToRGB == null)
00246             {
00247                 convertToRGB = true;
00248             }
00249         }
00250         ExitCodes result = (ExitCodes)NativeMethods.SaveRasterImage(OwnerContext.NativeContext,
00251             this.NativePointer, fileName, (int)RasterOutputFileTypes.JPG, quality, convertToRGB == true ? 1 :
00252             0);
00253         switch (result)
00254         {
00255             case ExitCodes.EXIT_SUCCESS:
00256                 break;
00257             case ExitCodes.ERR_CANNOT_RENDER:
00258                 throw new MuPDFException("An error occurred while rendering the image.", result);
00259             case ExitCodes.ERR_CANNOT_SAVE:
00260                 throw new MuPDFException("An error occurred while saving the image.", result);
00261             default:
00262                 throw new MuPDFException("Unknown error", result);
00263         }
00264     }
00265 }
00266 
```

```

00262 /// <param name="outputStream">The output <see cref="Stream"/>.</param>
00263 /// <param name="fileType">The image format.</param>
00264 /// <param name="convertToRGB">If this is <see langword="true"/>, the image is converted to the RGB
00265 colour space before being saved. If this is <see langword="false"/>, the image is saved in the same
00266 colour space as it was encoded in the document. If this is <see langword="null"/> (the default), the
00267 image is converted to RGB only if the target colour space does not support the colour space of the
00268 image.</param>
00269 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image or saving
00270 it.</exception>
00271     public void Write(Stream outputStream, RasterOutputFileTypes fileType, bool? convertToRGB =
00272 null)
00273     {
00274         if (this.disposedValue)
00275         {
00276             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
00277 been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00278         }
00279         if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.CMYK && (fileType ==
00280 RasterOutputFileTypes.PNG || fileType == RasterOutputFileTypes.PNM))
00281         {
00282             if (convertToRGB == false)
00283             {
00284                 throw new ArgumentException("Images in CMYK colour space cannot be exported to PNG
00285 or PNM files!");
00286             }
00287             else if (convertToRGB == null)
00288             {
00289                 convertToRGB = true;
00290             }
00291             else if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.Separation && fileType !=
00292 RasterOutputFileTypes.PSD)
00293             {
00294                 if (convertToRGB == false)
00295                 {
00296                     throw new ArgumentException("Images in Separation colour space can only be
00297 exported in PSD format!");
00298                 }
00299                 else if (convertToRGB == null)
00300                 {
00301                     convertToRGB = true;
00302                 }
00303             }
00304             IntPtr outputBuffer = IntPtr.Zero;
00305             IntPtr outputData = IntPtr.Zero;
00306             ulong outputDataLength = 0;
00307
00308             ExitCodes result = (ExitCodes)NativeMethods.WriteRasterImage(OwnerContext.NativeContext,
00309 this.NativePointer, (int)fileType, 90, ref outputBuffer, ref outputData, ref outputDataLength,
00310 convertToRGB == true ? 1 : 0);
00311
00312             switch (result)
00313             {
00314                 case ExitCodes.EXIT_SUCCESS:
00315                     break;
00316                 case ExitCodes.ERR_CANNOT_RENDER:
00317                     throw new MuPDFException("Cannot render page", result);
00318                 case ExitCodes.ERR_CANNOT_CREATE_BUFFER:
00319                     throw new MuPDFException("Cannot create the output buffer", result);
00320                 default:
00321                     throw new MuPDFException("Unknown error", result);
00322             }
00323
00324             byte[] buffer = new byte[1024];
00325
00326             while (outputDataLength > 0)
00327             {
00328                 int bytesToCopy = (int)Math.Min(buffer.Length, (long)outputDataLength);
00329                 Marshal.Copy(outputData, buffer, 0, bytesToCopy);
00330                 outputData = IntPtr.Add(outputData, bytesToCopy);
00331                 outputStream.Write(buffer, 0, bytesToCopy);
00332                 outputDataLength -= (ulong)bytesToCopy;
00333             }
00334
00335             NativeMethods.DisposeBuffer(OwnerContext.NativeContext, outputBuffer);
00336         }
00337
00338 /// <summary>
00339 /// Write the image to a <see cref="Stream"/> in JPEG format.
00340 /// </summary>
00341 /// <param name="outputStream">The output <see cref="Stream"/>.</param>
00342 /// <param name="quality">The quality of the JPEG image (from 0 to 100).</param>
00343 /// <param name="convertToRGB">If this is <see langword="true"/>, the image is converted to the RGB
00344 colour space before being saved. If this is <see langword="false"/>, the image is saved in the same
00345 colour space as it was encoded in the document. If this is <see langword="null"/> (the default), the

```

```

    image is converted to RGB only if the target colour space does not support the colour space of the
    image.</param>
00334 /// <exception cref="ArgumentOutOfRangeException">Thrown if the quality value is < 0 or >
0</exception>
00335 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image or saving
it.</exception>
00336     public void WriteAsJPEG(Stream outputStream, int quality, bool? convertToRGB = null)
00337     {
00338         if (this.disposedValue)
00339         {
00340             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00341         }
00342
00343         if (quality < 0 || quality > 100)
00344         {
00345             throw new ArgumentOutOfRangeException(nameof(quality), quality, "The JPEG quality must
range between 0 and 100 (inclusive)!");
00346         }
00347
00348         if (this.ColorSpace.RootColorSpace.Type == ColorSpaceType.Separation)
00349         {
00350             if (convertToRGB == false)
00351             {
00352                 throw new ArgumentException("Images in Separation colour space can only be
exported in PSD format!");
00353             }
00354             else if (convertToRGB == null)
00355             {
00356                 convertToRGB = true;
00357             }
00358         }
00359
00360         IntPtr outputBuffer = IntPtr.Zero;
00361         IntPtr outputData = IntPtr.Zero;
00362         ulong outputDataLength = 0;
00363
00364         ExitCodes result = (ExitCodes)NativeMethods.WriteRasterImage(OwnerContext.NativeContext,
this.NativePointer, (int)RasterOutputFileTypes.JPEG, quality, ref outputBuffer, ref outputData, ref
outputDataLength, convertToRGB == true ? 1 : 0);
00365
00366         switch (result)
00367         {
00368             case ExitCodes.EXIT_SUCCESS:
00369                 break;
00370             case ExitCodes.ERR_CANNOT_RENDER:
00371                 throw new MuPDFException("Cannot render page", result);
00372             case ExitCodes.ERR_CANNOT_CREATE_BUFFER:
00373                 throw new MuPDFException("Cannot create the output buffer", result);
00374             default:
00375                 throw new MuPDFException("Unknown error", result);
00376         }
00377
00378         byte[] buffer = new byte[1024];
00379
00380         while (outputDataLength > 0)
00381         {
00382             int bytesToCopy = (int)Math.Min(buffer.Length, (long)outputDataLength);
00383             Marshal.Copy(outputData, buffer, 0, bytesToCopy);
00384             outputData = IntPtr.Add(outputData, bytesToCopy);
00385             outputStream.Write(buffer, 0, bytesToCopy);
00386             outputDataLength -= (ulong)bytesToCopy;
00387         }
00388
00389         NativeMethods.DisposeBuffer(OwnerContext.NativeContext, outputBuffer);
00390     }
00391
00392 /// <summary>
00393 /// Get a byte representation of the image pixels.
00394 /// </summary>
00395 /// <returns>A byte representation of the image pixels in the specified pixel format.</returns>
00396 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image.</exception>
00397     public unsafe byte[] GetBytes(PixelFormats pixelFormat)
00398     {
00399         if (this.disposedValue)
00400         {
00401             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00402         }
00403
00404         IntPtr pixmap = IntPtr.Zero;
00405         IntPtr samples = IntPtr.Zero;
00406         int sampleCount = 0;
00407
00408         ExitCodes result = (ExitCodes)NativeMethods.LoadPixmapRGB(OwnerContext.NativeContext,
this.NativePointer, (int)pixelFormat, ref pixmap, ref samples, ref sampleCount);
00409

```

```

00410         switch (result)
00411     {
00412         case ExitCodes.EXIT_SUCCESS:
00413             break;
00414         case ExitCodes.ERR_CANNOT_RENDER:
00415             throw new MuPDFException("Cannot render page", result);
00416         default:
00417             throw new MuPDFException("Unknown error", result);
00418     }
00419
00420     bool renderedHasAlpha = sampleCount / (this.Width * this.Height) == 4;
00421
00422     byte[] tbr;
00423
00424     if (!renderedHasAlpha && (pixelFormat == PixelFormats.RGBA || pixelFormat ==
00425     PixelFormats.BGRA))
00426     {
00427         tbr = new byte[this.Width * this.Height * 4];
00428
00429         byte* renderedPixels = (byte*)samples;
00430
00431         for (int i = 0; i < this.Width * this.Height; i++)
00432         {
00433             tbr[i * 4] = renderedPixels[i * 3];
00434             tbr[i * 4 + 1] = renderedPixels[i * 3 + 1];
00435             tbr[i * 4 + 2] = renderedPixels[i * 3 + 2];
00436             tbr[i * 4 + 3] = 255;
00437         }
00438     else
00439     {
00440         tbr = new byte[sampleCount];
00441         fixed (byte* ptr = tbr)
00442         {
00443             Buffer.MemoryCopy((byte*)samples, ptr, sampleCount, sampleCount);
00444         }
00445     }
00446
00447     NativeMethods.DisposePixmap(OwnerContext.NativeContext, pixmap);
00448
00449     return tbr;
00450 }
00451
00452 /// <summary>
00453 /// Get a byte representation of the image pixels.
00454 /// </summary>
00455 /// <returns>A byte representation of the image pixels, in the native image colour space.</returns>
00456 /// <exception cref="MuPDFException">Thrown if an error occurs while rendering the image.</exception>
00457     public unsafe byte[] GetBytes()
00458     {
00459         if (this.disposedValue)
00460         {
00461             throw new ObjectDisposedException("MuPDFImage", "The MuPDFImage object has already
00462             been disposed! Maybe you disposed the MuPDFStructuredTextPage that contained it?");
00463         }
00464
00465         IntPtr pixmap = IntPtr.Zero;
00466         IntPtr samples = IntPtr.Zero;
00467         int sampleCount = 0;
00468
00469         ExitCodes result = (ExitCodes)NativeMethods.LoadPixmap(OwnerContext.NativeContext,
00470             this.NativePointer, ref pixmap, ref samples, ref sampleCount);
00471
00472         switch (result)
00473     {
00474         case ExitCodes.EXIT_SUCCESS:
00475             break;
00476         case ExitCodes.ERR_CANNOT_RENDER:
00477             throw new MuPDFException("Cannot render page", result);
00478         default:
00479             throw new MuPDFException("Unknown error", result);
00480     }
00481
00482     byte[] tbr = new byte[sampleCount];
00483
00484     fixed (byte* ptr = tbr)
00485     {
00486         Buffer.MemoryCopy((byte*)samples, ptr, sampleCount, sampleCount);
00487     }
00488
00489     NativeMethods.DisposePixmap(OwnerContext.NativeContext, pixmap);
00490
00491     return tbr;
00492 }
00493
00494 /// <inheritdoc>
00495     protected virtual void Dispose(bool disposing)

```

```

00494         {
00495             if (!disposedValue)
00496             {
00497                 if (OwnerContext.disposedValue)
00498                 {
00499                     throw new LifetimeManagementException<MuPDFImage, MuPDFContext>(this,
00500                         OwnerContext, this.NativePointer, OwnerContext.NativeContext);
00501                 }
00502             NativeMethods.DisposeImage(OwnerContext.NativeContext, this.NativePointer);
00503             disposedValue = true;
00504         }
00505     }
00506 }
00507 /// <summary>
00508 /// Dispose the <see cref="MuPDFImage"/>.
00509 /// </summary>
00510     ~MuPDFImage()
00511     {
00512         Dispose(disposing: false);
00513     }
00514
00515 /// <inheritdoc/>
00516     public void Dispose()
00517     {
00518         Dispose(disposing: true);
00519         GC.SuppressFinalize(this);
00520     }
00521 }
00522 }
```

8.12 MuPDFLinks.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2024 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections;
00020 using System.Collections.Generic;
00021 using System.Text;
00022
00023 namespace MuPDFCore
00024 {
00025 /// <summary>
00026 /// A collection of <see cref="MuPDFLink"/> objects.
00027 /// </summary>
00028     public class MuPDFLinks : IReadOnlyList<MuPDFLink>, IDisposable
00029     {
00030         private bool disposedValue;
00031
00032 /// <inheritdoc/>
00033         public MuPDFLink this[int index] => Contents[index];
00034
00035 /// <inheritdoc/>
00036         public int Count => Contents.Length;
00037
00038         internal MuPDFPage OwnerPage { get; }
00039         private IntPtr FirstLinkPointer { get; }
00040
00041         internal MuPDFLink[] Contents { get; }
00042
00043         internal unsafe MuPDFLinks(MuPDFPage ownerPage)
00044         {
00045             this.OwnerPage = ownerPage;
00046
00047             IntPtr firstLink = IntPtr.Zero;
00048             int countLinks =
NativeMethods.CountLinks(ownerPage.OwnerDocument.OwnerContext.NativeContext, ownerPage.NativePage, ref
firstLink);
```

```

00049             if (countLinks > 0)
00050             {
00051                 this.FirstLinkPointer = firstLink;
00052                 int[] uriLengths = new int[countLinks];
00053                 IntPtr[] linkPointers = new IntPtr[countLinks];
00054
00055                 fixed (int* uriLengthsPtr = uriLengths)
00056                 fixed (IntPtr* linkPointersPtr = linkPointers)
00057                 {
00058                     NativeMethods.LoadLinks(firstLink, (IntPtr)linkPointersPtr,
00059                     (IntPtr)uriLengthsPtr);
00060                 }
00061
00062                 this.Contents = new MuPDFLink[countLinks];
00063
00064                 for (int i = 0; i < countLinks; i++)
00065                 {
00066                     this.Contents[i] = new MuPDFLink(this, linkPointers[i], uriLengths[i]);
00067                 }
00068             }
00069         else
00070         {
00071             this.Contents = new MuPDFLink[0];
00072             this.FirstLinkPointer = IntPtr.Zero;
00073         }
00074     }
00075
00076 /// <inheritdoc/>
00077     public IEnumarator<MuPDFLink> GetEnumarator()
00078     {
00079         return ((IEnumerable<MuPDFLink>)Contents).GetEnumarator();
00080     }
00081
00082     IEnumarator IEnumerable.GetEnumerator()
00083     {
00084         return Contents.GetEnumerator();
00085     }
00086
00087 /// <inheritdoc/>
00088     protected virtual void Dispose(bool disposing)
00089     {
00090         if (!disposedValue)
00091         {
00092             if (this.FirstLinkPointer != IntPtr.Zero)
00093             {
00094                 NativeMethods.DisposeLinks(this.OwnerPage.OwnerDocument.OwnerContext.NativeContext,
00095                 this.FirstLinkPointer);
00096             }
00097             disposedValue = true;
00098         }
00099     }
00100 /// <inheritdoc/>
00101     ~MuPDFLinks()
00102     {
00103         Dispose(disposing: false);
00104     }
00105
00106 /// <summary>
00107 /// <inheritdoc/>
00108 /// </summary>
00109     public void Dispose()
00110     {
00111         Dispose(disposing: true);
00112         GC.SuppressFinalize(this);
00113     }
00114 }
00115
00116 /// <summary>
00117 /// Represents a link within a MuPDF document.
00118 /// </summary>
00119     public class MuPDFLink
00120     {
00121         /// <summary>
00122         /// The area on which users can click to activate the link.
00123         /// </summary>
00124         public Rectangle ActiveArea { get; }
00125
00126         /// <summary>
00127         /// The link destination.
00128         /// </summary>
00129         public MuPDFLinkDestination Destination { get; }
00130
00131         /// <summary>
00132         /// Whether the link is visible or hidden (e.g., because it is part of a hidden optional content

```

```
group).
00133 /// <summary>
00134     public bool IsVisible => this.OwnerCollection.OwnerPage.OwnerDocument.NativePDFDocument ==
IntPtr.Zero ||
NativeMethods.IsLinkHidden(this.OwnerCollection.OwnerPage.OwnerDocument.OwnerContext.NativeContext,
"View", this.NativeLink) == 0;
00135
00136     private string Uri { get; }
00137     internal IntPtr NativeLink { get; }
00138     internal MuPDFLinks OwnerCollection { get; }
00139
00140     internal unsafe MuPDFLink(MuPDFLinks ownerCollection, IntPtr linkPointer, int uriLength)
00141     {
00142         this.OwnerCollection = ownerCollection;
00143         float x0 = 0;
00144         float x1 = 0;
00145         float y0 = 0;
00146         float y1 = 0;
00147
00148         byte[] uriBytes = new byte[uriLength];
00149
00150         int isExternal = -1;
00151         int isSetOCGState = -1;
00152
00153         int destinationType = -1;
00154         float x = 0;
00155         float y = 0;
00156         float w = 0;
00157         float h = 0;
00158         float zoom = 0;
00159         int page = -1;
00160         int chapter = -1;
00161
00162         fixed (byte* uriPtr = uriBytes)
00163         {
00164
NativeMethods.LoadLink(ownerCollection.OwnerPage.OwnerDocument.OwnerContext.NativeContext,
ownerCollection.OwnerPage.OwnerDocument.NativeDocument, linkPointer, uriLength,
ownerCollection.OwnerPage.OwnerDocument.NativePDFDocument != IntPtr.Zero ? 1 : 0, ref x0, ref y0,
ref x1, ref y1, (IntPtr)uriPtr, ref isExternal, ref isSetOCGState, ref destinationType, ref x, ref y,
ref w, ref h, ref zoom, ref chapter, ref page);
00165     }
00166
00167     this.ActiveArea = new Rectangle(x0, y0, x1, y1);
00168
00169     string uri = Encoding.UTF8.GetString(uriBytes);
00170
00171     this.Uri = uri;
00172     this.NativeLink = linkPointer;
00173
00174     if (isExternal != 0)
00175     {
00176         this.Destination = new MuPDFExternalLinkDestination(this, uri);
00177     }
00178     else if (isSetOCGState != 0)
00179     {
00180         this.Destination = new MuPDFSetOCGStateLinkDestination(this);
00181     }
00182     else
00183     {
00184         this.Destination = new MuPDFInternalLinkDestination(this, page, chapter, x, y, w, h,
zoom, destinationType);
00185     }
00186 }
00187 }
00188
00189 /// <summary>
00190 /// Represents a generic link destination.
00191 /// </summary>
00192     public abstract class MuPDFLinkDestination
00193     {
00194     /// <summary>
00195     /// Defines the types of link destinations.
00196     /// </summary>
00197         public enum DestinationType
00198         {
00199     /// <summary>
00200     /// An internal link.
00201     /// </summary>
00202         Internal,
00203
00204     /// <summary>
00205     /// An link to an external resource (e.g., a website).
00206     /// </summary>
00207         External,
00208
00209     /// <summary>
```

```

00210 /// A link whose effect is to change the visibility state of some optional content groups (also known
00211 /// as layers).
00212     /// </summary>
00213     SetOCGState
00214 }
00215 /// <summary>
00216 /// The type of link destination.
00217 /// </summary>
00218     public abstract DestinationType Type { get; }
00219
00220     internal MuPDFLink OwnerLink;
00221
00222     internal MuPDFLinkDestination(MuPDFLink ownerLink)
00223     {
00224         this.OwnerLink = ownerLink;
00225     }
00226 }
00227
00228 /// <summary>
00229 /// A link destination to an external resource (e.g., a website).
00230 /// </summary>
00231     public class MuPDFExternalLinkDestination : MuPDFLinkDestination
00232     {
00233     /// <summary>
00234     /// The Uri of the external resource.
00235     /// </summary>
00236     public string Uri { get; }
00237
00238 /// <inheritDoc/>
00239     public override DestinationType Type => DestinationType.External;
00240
00241     internal MuPDFExternalLinkDestination(MuPDFLink ownerLink, string uri) : base(ownerLink)
00242     {
00243         this.Uri = uri;
00244     }
00245 }
00246
00247 /// <summary>
00248 /// A destination for a link whose effect is to change the visibility state of some optional content
00249 /// groups (also known as layers).
00250 /// </summary>
00251     public class MuPDFSetOCGStateLinkDestination : MuPDFLinkDestination
00252     {
00253     /// <inheritDoc/>
00254     public override DestinationType Type => DestinationType.SetOCGState;
00255
00256     internal MuPDFSetOCGStateLinkDestination(MuPDFLink ownerLink) : base(ownerLink) { }
00257
00258 /// <summary>
00259 /// Activate the link destination, thus showing/hiding optional content groups as necessary.
00260     public void Activate()
00261     {
00262
00263     NativeMethods.ActivateLinkSetOCGState(this.OwnerLink.OwnerCollection.OwnerPage.OwnerDocument.OwnerContext.NativeContext,
00264     this.OwnerLink.OwnerCollection.OwnerPage.OwnerDocument.NativePDFDocument, this.OwnerLink.NativeLink);
00265
00266     this.OwnerLink.OwnerCollection.OwnerPage.OwnerDocument.ClearCache();
00267 }
00268
00269 /// <summary>
00270 /// An internal link destination.
00271 /// </summary>
00272     public class MuPDFInternalLinkDestination : MuPDFLinkDestination
00273     {
00274     /// <summary>
00275     /// Defines internal link destination types.
00276     /// </summary>
00277     public enum InternalDestinationType
00278     {
00279     /// <summary>
00280     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that it
00281     /// completely fits within the view area.
00282     /// </summary>
00283     Fit,
00284
00285     /// <summary>
00286     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that its
00287     /// bounding box completely fits within the view area.
00288     /// </summary>
00289     FitBoundingBox,
00290
00291     /// <summary>
00292     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that the page
00293     /// width completely fits within the view area.
00294     /// </summary>

```

```
00290         FitWidth,
00291
00292     /// <summary>
00293     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that the
00294     /// bounding box width completely fits within the view area.
00295     /// </summary>
00296     FitBoundingBoxWidth,
00297
00298     /// <summary>
00299     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that the page
00300     /// height completely fits within the view area.
00301     /// </summary>
00302     FitHeight,
00303
00304     /// <summary>
00305     /// Display the target <see cref="Page"/>, with an appropriate magnification factor so that the
00306     /// bounding box height completely fits within the view area.
00307     /// </summary>
00308     /// Display the target <see cref="Page"/>, with an appropriate magnification factor and traslation, so
00309     /// that the rectangle specified by {<see cref="X"/>, <see cref="Y"/>, <see cref="Width"/>, <see
00310     /// cref="Height"/>} completely fits within the viewing area.
00311     FitRectangle,
00312
00313     /// <summary>
00314     /// Display the target <see cref="Page"/> with {<see cref="X"/>, <see cref="Y"/>} on the top-left
00315     /// corner and the specified <see cref="Zoom"/> (which can be <c>0</c> - in this case, the zoom factor
00316     /// should be unchanged).
00317     XYZoom
00318
00319     public override DestinationType Type => DestinationType.Internal;
00320
00321     /// <summary>
00322     /// Locations within the document are referred to in terms of chapter and page, rather than just a
00323     /// page number. For some documents (such as epub documents with large numbers of pages broken into many
00324     /// chapters) this can make navigation much faster as only the required chapter needs to be decoded at a
00325     /// time.
00326     /// </summary>
00327     /// The page number of an internal link, relative to the specified <see cref="Chapter"/>, or -1 for
00328     /// external links or links with no destination.
00329     public int Chapter { get; }
00330
00331     private int? CachedPageNumber = null;
00332
00333     /// <summary>
00334     /// The overall page number of an internal link. This is determined on first access, and it might
00335     /// cause a large number of chapters to be laid out to determine it.
00336     public int PageNumber
00337     {
00338         get
00339         {
00340             if (CachedPageNumber == null)
00341             {
00342                 CachedPageNumber =
00343                     NativeMethods.GetPageNumber(this.OwnerLink.OwnerCollection.OwnerPage.OwnerDocument.OwnerContext.NativeContext,
00344                     this.OwnerLink.OwnerCollection.OwnerPage.OwnerDocument.NativeDocument, this.Chapter, this.Page);
00345
00346             }
00347         }
00348
00349     /// <summary>
00350     /// X coordinate of the link target on the specified page.
00351     /// </summary>
00352     public float X { get; }
00353
00354     /// <summary>
00355     /// Y coordinate of the link target on the specified page.
00356     /// </summary>
00357     public float Y { get; }
00358
00359     /// <summary>
00360     /// Width of the link target on the specified page.
00361     /// </summary>
00362     public float Width { get; }
```

```

00363
00364     /// <summary>
00365     /// Height of the link target on the specified page.
00366     /// </summary>
00367     public float Height { get; }
00368
00369     /// <summary>
00370     /// Target magnification factor.
00371     /// </summary>
00372     public float Zoom { get; }
00373
00374     /// <summary>
00375     /// The type of internal link destination.
00376     /// </summary>
00377     public InternalDestinationType InternalType { get; }
00378
00379     internal MuPDFInternalLinkDestination(MuPDFLink ownerLink, int page, int chapter, float x,
00380     float y, float w, float h, float zoom, int type) : base(ownerLink)
00380     {
00381         this.Page = page;
00382         this.Chapter = chapter;
00383         this.X = x;
00384         this.Y = y;
00385         this.Width = w;
00386         this.Height = h;
00387         this.Zoom = zoom;
00388         this.InternalType = (InternalDestinationType)type;
00389     }
00390 }
00391 }
```

8.13 MuPDFMultiThreadedPageRenderer.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections.Generic;
00020 using System.Linq;
00021 using System.Runtime.InteropServices;
00022 using System.Threading;
00023
00024 namespace MuPDFCore
00025 {
00026     /// <summary>
00027     /// A reusable thread to render part of an image.
00028     /// </summary>
00029     internal class RenderingThread : IDisposable
00030     {
00031         /// <summary>
00032         /// A class to hold the data used internally by the rendering thread.
00033         /// </summary>
00034         private class RenderData
00035         {
00036             public IntPtr Context;
00037             public MuPDFDisplayList DisplayList;
00038             public Rectangle Region;
00039             public float Zoom;
00040             public IntPtr PixelStorage;
00041             public Rectangle PageBounds;
00042             public PixelFormats PixelFormat;
00043             public bool ClipToPageBounds;
00044         }
00045
00046         /// <summary>
00047         /// A lock object to prevent race conditions.
00048         /// </summary>
00049         private readonly object RenderDataLock;
00050 }
```

```
00051 /// <summary>
00052 /// The data used internally by the rendering thread.
00053 /// </summary>
00054     private readonly RenderData CurrentRenderData;
00055
00056 /// <summary>
00057 /// The actual thread that does the rendering.
00058 /// </summary>
00059     private readonly Thread Thread;
00060
00061 /// <summary>
00062 /// An <see cref="EventWaitHandle"/> that is set by the <see cref="Thread"/> when it finished
00063 /// rendering.
00064     private readonly EventWaitHandle SignalFromThread;
00065
00066 /// <summary>
00067 /// An <see cref="EventWaitHandle"/> that signals to the <see cref="Thread"/> that it should start
00068 /// rendering.
00069     private readonly EventWaitHandle SignalToThread;
00070
00071 /// <summary>
00072 /// An <see cref="EventWaitHandle"/> that signals that the object is being disposed and all activity
00073 /// should cease.
00074     private readonly EventWaitHandle DisposeSignal;
00075
00076 /// <summary>
00077 /// A pointer to a <see cref="Cookie"/> object that can be used to monitor the progress of the
00078 /// rendering or to abort it.
00079     private readonly IntPtr Cookie;
00080
00081 /// <summary>
00082 /// Performs the actual rendering.
00083 /// </summary>
00084     private void RenderAction()
00085     {
00086         ExitCodes result =
00087             (ExitCodes)NativeMethods.RenderSubDisplayList(this.CurrentRenderData.Context,
00088                 this.CurrentRenderData.DisplayList.NativeDisplayList, this.CurrentRenderData.Region.X0,
00089                 this.CurrentRenderData.Region.Y0, this.CurrentRenderData.Region.X1, this.CurrentRenderData.Region.Y1,
00090                 this.CurrentRenderData.Zoom, (int)this.CurrentRenderData.PixelFormat,
00091                 this.CurrentRenderData.PixelStorage, Cookie);
00092
00093         switch (result)
00094         {
00095             case ExitCodes.EXIT_SUCCESS:
00096                 break;
00097             case ExitCodes.ERR_CANNOT_RENDER:
00098                 throw new MuPDFException("Cannot render page", result);
00099             default:
00100                 throw new MuPDFException("Unknown error", result);
00101         }
00102
00103         RoundedRectangle roundedRegion =
00104             this.CurrentRenderData.Region.Round(this.CurrentRenderData.Zoom);
00105         RoundedSize roundedSize = new RoundedSize(roundedRegion.Width, roundedRegion.Height);
00106
00107         if (this.CurrentRenderData.PixelFormat == PixelFormats.RGBA ||
00108             this.CurrentRenderData.PixelFormat == PixelFormats.BGRA)
00109             Utils.UnpremultiplyAlpha(this.CurrentRenderData.PixelStorage, roundedSize);
00110
00111         if (this.CurrentRenderData.ClipToPageBounds &&
00112             !this.CurrentRenderData.PageBounds.Contains(this.CurrentRenderData.DisplayList.Bounds.Intersect(this.CurrentRenderData.
00113             PageBounds))
00114             {
00115                 Utils.ClipImage(this.CurrentRenderData.PixelStorage, roundedSize,
00116                     this.CurrentRenderData.Region, this.CurrentRenderData.PageBounds, this.CurrentRenderData.PixelFormat);
00117             }
00118
00119     /// <summary>
00120     /// Create a new <see cref="RenderingThread"/> instance.
00121     /// </summary>
00122     public RenderingThread()
00123     {
00124         //Initialize fields
00125         SignalFromThread = new EventWaitHandle(false, EventResetMode.ManualReset);
00126         SignalToThread = new EventWaitHandle(false, EventResetMode.ManualReset);
00127         DisposeSignal = new EventWaitHandle(false, EventResetMode.ManualReset);
00128
00129         CurrentRenderData = new RenderData();
00130         RenderDataLock = new object();
00131     }
```

```

00125     //Allocate unmanaged memory to hold the cookie.
00126     Cookie = Marshal.AllocHGlobal(Marshal.SizeOf<MuPDFCore.Cookie>());
00127
00128     //Initialise the rendering thread.
00129     this.Thread = new Thread(() =>
00130     {
00131         EventWaitHandle[] handles = new EventWaitHandle[] { SignalToThread, DisposeSignal };
00132
00133         while (true)
00134         {
00135             int handle = EventWaitHandle.WaitAny(handles);
00136
00137             if (handle == 0)
00138             {
00139                 SignalToThread.Reset();
00140
00141                 lock (RenderDataLock)
00142                 {
00143                     this.RenderAction();
00144                 }
00145
00146                 SignalFromThread.Set();
00147             }
00148             else
00149             {
00150                 SignalFromThread.Set();
00151                 break;
00152             }
00153         }
00154     });
00155
00156     //Start the rendering thread.
00157     this.Thread.Start();
00158 }
00159
00160 /// <summary>
00161 /// Start rendering a region of a display list to the specified destination.
00162 /// </summary>
00163 /// <param name="context">The rendering context.</param>
00164 /// <param name="displayList">The native display list that should be rendered.</param>
00165 /// <param name="region">The region that should be rendered.</param>
00166 /// <param name="zoom">The scale at which the region will be rendered. This will determine the size
in pixel of the image.</param>
00167 /// <param name="pixelStorage">The address of the buffer where the pixel data will be written. There
must be enough space available to write the values for all the pixels, otherwise this will fail
catastrophically!</param>
00168 /// <param name="pixelFormat">The format of the pixel data.</param>
00169 /// <param name="pageBounds">The bounds of the page being rendered.</param>
00170 /// <param name="clipToPageBounds">A boolean value indicating whether the rendered image should be
clipped to the original page's bounds. This can be relevant if the page has been "cropped" by
altering its mediabox, but otherwise leaving the contents untouched.</param>
00171     public void Render(IntPtr context, MuPDFDisplayList displayList, Rectangle region, float zoom,
IntPtr pixelStorage, PixelFormats pixelFormat, Rectangle pageBounds, bool clipToPageBounds)
00172     {
00173         lock (RenderDataLock)
00174         {
00175             //Reset the cookie.
00176             unsafe
00177             {
00178                 Cookie* cookie = (Cookie*)Cookie;
00179
00180                 cookie->abort = 0;
00181                 cookie->errors = 0;
00182                 cookie->incomplete = 0;
00183                 cookie->progress = 0;
00184                 cookie->progress_max = 0;
00185             }
00186
00187             //Set up all the rendering data.
00188             CurrentRenderData.Context = context;
00189             CurrentRenderData.DisplayList = displayList;
00190             CurrentRenderData.Region = region;
00191             CurrentRenderData.Zoom = zoom;
00192             CurrentRenderData.PixelStorage = pixelStorage;
00193             CurrentRenderData.PixelFormat = pixelFormat;
00194             CurrentRenderData.PageBounds = pageBounds;
00195             CurrentRenderData.ClipToPageBounds = clipToPageBounds;
00196             SignalToThread.Set();
00197         }
00198     }
00199
00200 /// <summary>
00201 /// Wait until the current rendering operation finishes.
00202 /// </summary>
00203     public void WaitForRendering()
00204     {
00205         EventWaitHandle[] handles = new EventWaitHandle[] { SignalFromThread, DisposeSignal };

```

```
00206             int result = EventWaitHandle.WaitAny(handles);
00207             if (result == 0)
00208             {
00209                 SignalFromThread.Reset();
00210             }
00211         }
00212 
00213 /// <summary>
00214 /// Abort the current rendering operation.
00215 /// </summary>
00216     public void AbortRendering()
00217     {
00218         lock (RenderDataLock)
00219         {
00220             unsafe
00221             {
00222                 Cookie* cookie = (Cookie*)Cookie;
00223                 cookie->abort = 1;
00224             }
00225         }
00226     }
00227 
00228 /// <summary>
00229 /// Get the progress of the current rendering operation.
00230 /// </summary>
00231 /// <returns>A <see cref="RenderProgress.ThreadRenderProgress"/> object containing the progress of the
00232 /// current rendering operation.</returns>
00233     public RenderProgress.ThreadRenderProgress GetProgress()
00234     {
00235         int progress;
00236         ulong maxProgress;
00237 
00238         unsafe
00239         {
00240             Cookie* cookie = (Cookie*)Cookie;
00241 
00242             progress = cookie->progress;
00243             maxProgress = cookie->progress_max;
00244         }
00245 
00246         return new RenderProgress.ThreadRenderProgress(progress, maxProgress);
00247     }
00248 
00249     private bool disposedValue;
00250 
00251     protected virtual void Dispose(bool disposing)
00252     {
00253         if (!disposedValue)
00254         {
00255             if (disposing)
00256             {
00257                 DisposeSignal.Set();
00258                 Thread.Join();
00259             }
00260 
00261             Marshal.FreeHGlobal(Cookie);
00262 
00263             disposedValue = true;
00264         }
00265     }
00266 
00267     public void Dispose()
00268     {
00269         Dispose(disposing: true);
00270         GC.SuppressFinalize(this);
00271     }
00272 }
00273 /// <summary>
00274 /// A class that holds the necessary resources to render a page of a MuPDF document using multiple
00275 /// threads.
00276 /// </summary>
00277     public class MuPDFMultiThreadedPageRenderer : IDisposable
00278     {
00279         /// <summary>
00280         /// The display list that is rendered by this renderer.
00281         /// </summary>
00282         private readonly MuPDFDisplayList DisplayList;
00283 
00284         /// <summary>
00285         /// The cloned contexts that are used by the <see cref="RenderingThreads"/> to render the display
00286         /// list.
00287         /// </summary>
00288         private readonly MuPDFContext[] Contexts;
00289 
00290         /// <summary>
00291         /// The <see cref="RenderingThreads"/> that are in charge of the actual rendering.
00292 }
```

```
00290     /// </summary>
00291     private readonly RenderingThread[] RenderingThreads;
00292
00293     /// <summary>
00294     /// The bounds of the page being rendered.
00295     /// </summary>
00296     private readonly Rectangle PageBounds;
00297
00298     /// <summary>
00299     /// Whether the rendered images should be clipped to the bounds of the page being rendered.
00300     /// </summary>
00301     private readonly bool ClipToPageBounds;
00302
00303     /// <summary>
00304     /// The number of threads that are used to render the image.
00305     /// </summary>
00306     public int ThreadCount { get; }
00307
00308     /// <summary>
00309     /// If the document is an image, the horizontal resolution of the image. Otherwise, 72.
00310     /// </summary>
00311     internal double ImageXRes = double.NaN;
00312
00313     /// <summary>
00314     /// If the document is an image, the vertical resolution of the image. Otherwise, 72.
00315     /// </summary>
00316     internal double ImageYRes = double.NaN;
00317
00318     /// <summary>
00319     /// Create a new <see cref="MuPDFMultiThreadedPageRenderer"/> from a specified display list using the
00320     /// specified number of threads.
00321     /// </summary>
00322     /// <param name="context">The context that owns the document from which the display list was
00323     /// extracted.</param>
00324     /// <param name="displayList">The display list to render.</param>
00325     /// <param name="threadCount">The number of threads to use in the rendering. This must be
00326     /// factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <paramref
00327     /// name="threadCount"/> that satisfies this condition is used.</param>
00328     /// <param name="pageBounds">The bounds of the page being rendered.</param>
00329     /// <param name="clipToPageBounds">A boolean value indicating whether the rendered image should be
00330     /// clipped to the original page's bounds. This can be relevant if the page has been "cropped" by
00331     /// altering its mediabox, but otherwise leaving the contents untouched.</param>
00332     /// <param name="imageXRes">If the document is an image, the horizontal resolution of the image.
00333     /// Otherwise, 72.</param>
00334     /// <param name="imageYRes">If the document is an image, the vertical resolution of the image.
00335     /// Otherwise, 72.</param>
00336
00337     internal MuPDFMultiThreadedPageRenderer(MuPDFContext context, MuPDFDisplayList displayList,
00338     int threadCount, Rectangle pageBounds, bool clipToPageBounds, double imageXRes, double imageYRes)
00339     {
00340         threadCount = Utils.GetAcceptableNumber(threadCount);
00341
00342         this.ThreadCount = threadCount;
00343         this.DisplayList = displayList;
00344         this.PageBounds = pageBounds;
00345         this.ClipToPageBounds = clipToPageBounds;
00346
00347         this.ImageXRes = imageXRes;
00348         this.ImageYRes = imageYRes;
00349
00350         IntPtr[] contexts = new IntPtr[threadCount];
00351         GCHandle contextsHandle = GCHandle.Alloc(contexts, GCHandleType.Pinned);
00352
00353         try
00354         {
00355             ExitCodes result = (ExitCodes)NativeMethods.CloneContext(context.NativeContext,
00356             threadCount, contextsHandle.AddrOfPinnedObject());
00357
00358             switch (result)
00359             {
00360                 case ExitCodes.EXIT_SUCCESS:
00361                     break;
00362                 case ExitCodes.ERR_CANNOT_INIT_MUTEX:
00363                     throw new MuPDFException("Cannot initialize mutex objects", result);
00364                 case ExitCodes.ERR_CANNOT_CREATE_CONTEXT:
00365                     throw new MuPDFException("Cannot create master context", result);
00366                 case ExitCodes.ERR_CANNOT_CLONE_CONTEXT:
00367                     throw new MuPDFException("Cannot create context clones", result);
00368                 default:
00369                     throw new MuPDFException("Unknown error", result);
00370             }
00371
00372             Contexts = new MuPDFContext[threadCount];
00373             RenderingThreads = new RenderingThread[threadCount];
00374             for (int i = 0; i < threadCount; i++)
00375             {
00376                 Contexts[i] = new MuPDFContext(context, contexts[i]);
00377                 RenderingThreads[i] = new RenderingThread();
00378             }
00379         }
00380     }
```

```

00367         }
00368     }
00369     finally
00370     {
00371         contextsHandle.Free();
00372     }
00373 }
00374
00375 /// <summary>
00376 /// Render the specified region to an image of the specified size, split in a number of tiles equal to
00377 /// the number of threads used by this <see cref="MuPDFMultiThreadedPageRenderer"/>, without marshaling.
00378 /// This method will not return until all the rendering threads have finished.
00379 /// </summary>
00380 /// <param name="targetSize">The total size of the image that should be rendered.</param>
00381 /// <param name="region">The region in page units that should be rendered.</param>
00382 /// <param name="destinations">An array containing the addresses of the buffers where the rendered
00383 /// tiles will be written. There must be enough space available in each buffer to write the values for
00384 /// all the pixels of the tile, otherwise this will fail catastrophically!
00385 /// As long as the <paramref name="targetSize"/> is the same, the size in pixel of the tiles is
00386 /// guaranteed to also be the same.</param>
00387 /// <param name="pixelFormat">The format of the pixel data.</param>
00388 public void Render(RoundedSize targetSize, Rectangle region, IntPtr[] destinations,
00389     PixelFormats pixelFormat)
00390 {
00391     if (destinations.Length != Contexts.Length)
00392     {
00393         throw new ArgumentOutOfRangeException(nameof(destinations), destinations.Length, "The
00394             number of destinations must be equal to the number of rendering threads!");
00395     }
00396
00397     RoundedRectangle[] targets = targetSize.Split(destinations.Length);
00398
00399     float zoomX = targetSize.Width / region.Width;
00400     float zoomY = targetSize.Height / region.Height;
00401
00402     float zoom = (float)Math.Sqrt(zoomX * zoomY);
00403
00404     Rectangle actualPageArea = new Rectangle(region.X0, region.Y0, region.X0 +
00405         targetSize.Width / zoom, region.Y0 + targetSize.Height / zoom);
00406
00407     Rectangle[] origins = actualPageArea.Split(destinations.Length);
00408
00409     //Make sure that each tile has the expected size in pixel, rounding errors
00410     //notwithstanding.
00411     for (int i = 0; i < origins.Length; i++)
00412     {
00413         int countBlanks = 0;
00414         RoundedRectangle roundedOrigin = origins[i].Round(zoom);
00415         while (roundedOrigin.Width != targets[i].Width || roundedOrigin.Height !=
00416             targets[i].Height)
00417         {
00418             RoundedRectangle oldRoundedOrigin = roundedOrigin;
00419
00420             if (roundedOrigin.Width > targets[i].Width)
00421             {
00422                 if (origins[i].X0 > actualPageArea.X0)
00423                 {
00424                     origins[i] = new Rectangle(origins[i].X0 + 0.5 / zoom, origins[i].Y0,
00425                         origins[i].X1, origins[i].Y1);
00426                 }
00427                 else
00428                 {
00429                     origins[i] = new Rectangle(origins[i].X0, origins[i].Y0, origins[i].X1 -
00430                         0.5 / zoom, origins[i].Y1);
00431                 }
00432
00433             if (roundedOrigin.Height > targets[i].Height)
00434             {
00435                 if (origins[i].Y0 > actualPageArea.Y0)
00436                 {
00437                     origins[i] = new Rectangle(origins[i].X0, origins[i].Y0 + 0.5 / zoom,
00438                         origins[i].X1, origins[i].Y1);
00439                 }
00440             }
00441         }
00442     }
00443 }
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988
01989
01990
01991
01992
01993
01994
01995
01996
01997
01998
01999
02000
02001
02002
02003
02004
02005
02006
02007
02008
02009
02010
02011
02012
02013
02014
02015
02016
02017
02018
02019
02020
02021
02022
02023
02024
02025
02026
02027
02028
02029
02030
02031
02032
02033
02034
02035
02036
02037
02038
02039
02040
02041
02042
02043
02044
02045
02046
02047
02048
02049
02050
02051
02052
02053
02054
02055
02056
02057
02058
02059
02060
02061
02062
02063
02064
02065
02066
02067
02068
02069
02070
02071
02072
02073
02074
02075
02076
02077
02078
02079
02080
02081
02082
02083
02084
02085
02086
02087
02088
02089
02090
02091
02092
02093
02094
02095
02096
02097
02098
0
```

```

00439             }
00440             else
00441             {
00442                 origins[i] = new Rectangle(origins[i].X0, origins[i].Y0, origins[i].X1,
00443                                         origins[i].Y1 - 0.5 / zoom);
00444             }
00445         }
00446         else if (roundedOrigin.Height < targets[i].Height)
00447         {
00448             if (origins[i].X1 < actualPageArea.X1)
00449             {
00450                 origins[i] = new Rectangle(origins[i].X0, origins[i].Y0, origins[i].X1,
00451                                         origins[i].Y1 + 0.5 / zoom);
00452             }
00453             else
00454             {
00455                 origins[i] = new Rectangle(origins[i].X0, origins[i].Y0 - 0.5 / zoom,
00456                                         origins[i].X1, origins[i].Y1);
00457             }
00458         }
00459         roundedOrigin = origins[i].Round(zoom);
00460
00461         if (roundedOrigin.Width == oldRoundedOrigin.Width && roundedOrigin.Height ==
00462             oldRoundedOrigin.Height && (roundedOrigin.Width != targets[i].Width || roundedOrigin.Height !=
00463             targets[i].Height))
00464         {
00465             countBlanks++;
00466         }
00467         if (countBlanks >= 100)
00468         {
00469             //It seems that we can't coerce the expected size and the actual size to be
00470             the same. Give up.
00471             return;
00472         }
00473     }
00474
00475     //Start each rendering thread.
00476     for (int i = 0; i < destinations.Length; i++)
00477     {
00478         double dzoom = zoom;
00479         Rectangle origin = origins[i];
00480         if (this.ImageXRes != 72 || this.ImageYRes != 72)
00481         {
00482             dzoom *= Math.Sqrt(this.ImageXRes * this.ImageYRes) / 72;
00483             origin = new Rectangle(origin.X0 * 72 / this.ImageXRes, origin.Y0 * 72 /
00484             this.ImageYRes, origin.X1 * 72 / this.ImageXRes, origin.Y1 * 72 / this.ImageYRes);
00485         }
00486         RenderingThreads[i].Render(Contexts[i].NativeContext, DisplayList, origin,
00487         (float)dzoom, destinations[i], pixelFormat, this.PageBounds, ClipToPageBounds);
00488     }
00489
00490     //Wait until all the rendering threads have finished.
00491     for (int i = 0; i < destinations.Length; i++)
00492     {
00493         RenderingThreads[i].WaitForRendering();
00494     }
00495
00496 /// <summary>
00497 /// Gets an element from a collection of <see cref="System.Span{T}">Span</see>&lt;see
00498 /// cref="byte"/>&gt;;
00499 /// </summary>
00500 /// <param name="index">The index of the element to get.</param>
00501 /// <returns>An element from a collection of <see cref="System.Span{T}">Span</see>&lt;see
00502 /// cref="byte"/>&gt;;</returns>
00503     public delegate Span<byte> GetSpanItem(int index);
00504
00505 /// <summary>
00506 /// Render the specified region to an image of the specified size, split in a number of tiles equal to
00507 /// the number of threads used by this <see cref="MuPDFMultiThreadedPageRenderer"/>, without marshaling.
00508 /// This method will not return until all the rendering threads have finished.
00509 /// Since creating an array of <see cref="Span{T}">Span</see> is not allowed, this method returns a delegate
00510 /// that accepts an integer parameter (representing the index of the span in the "array") and returns the
00511 /// <see cref="Span{T}">Span</see> corresponding to that index.
00512
00513 /// <summary>
00514 /// <param name="targetSize">The total size of the image that should be rendered.</param>
00515 /// <param name="region">The region in page units that should be rendered.</param>
00516 /// <param name="disposables">A collection of <see cref="IDisposable"/>s that can be used to free the
00517 /// memory where the rendered tiles are stored. You should keep track of these and dispose them when you
00518 /// have finished working with the images.</param>
00519 /// <param name="pixelFormat">The format of the pixel data.</param>
00520 /// <returns>A delegate that accepts an integer parameter (representing the index of the span in the
00521 /// "array") and returns the <see cref="Span{T}">Span</see> corresponding to that index.</returns>

```

```
00509     public GetSpanItem Render(RoundedSize targetSize, Rectangle region, out IDisposable[]
00510         disposables, PixelFormats pixelFormat)
00511     {
00512         RoundedRectangle[] targets = targetSize.Split(this.ThreadCount);
00513
00514         IntPtr[] destinations = new IntPtr[targets.Length];
00515         disposables = new IDisposable[targets.Length];
00516
00517         bool hasAlpha = pixelFormat == PixelFormats.RGBA || pixelFormat == PixelFormats.BGRA;
00518
00519         int[] sizes = new int[destinations.Length];
00520
00521         for (int i = 0; i < targets.Length; i++)
00522         {
00523             int allocSize = targets[i].Width * targets[i].Height * (hasAlpha ? 4 : 3);
00524
00525             sizes[i] = allocSize;
00526             destinations[i] = Marshal.AllocHGlobal(allocSize);
00527             disposables[i] = new DisposableIntPtr(destinations[i], allocSize);
00528         }
00529
00530         this.Render(targetSize, region, destinations, pixelFormat);
00531
00532         return i =>
00533     {
00534         unsafe
00535         {
00536             return new Span<byte>((void*)destinations[i], sizes[i]);
00537         };
00538     }
00539
00540 /// <summary>
00541 /// Signal to the rendering threads that they should abort rendering as soon as possible.
00542 /// </summary>
00543     public void Abort()
00544     {
00545         for (int i = 0; i < RenderingThreads.Length; i++)
00546         {
00547             RenderingThreads[i].AbortRendering();
00548         }
00549     }
00550
00551 /// <summary>
00552 /// Get the current rendering progress of all the threads.
00553 /// </summary>
00554 /// <returns>A <see cref="RenderProgress"/> object containing the rendering progress of all the
00555     threads.</returns>
00556     public RenderProgress GetProgress()
00557     {
00558         return new RenderProgress((from el in RenderingThreads select
00559             el.GetProgress()).ToArray());
00560     }
00561
00562 //<inheritDoc>
00563     protected virtual void Dispose(bool disposing)
00564     {
00565         if (!disposedValue)
00566         {
00567             if (disposing)
00568             {
00569                 Abort();
00570
00571                 if (RenderingThreads != null)
00572                 {
00573                     for (int i = 0; i < RenderingThreads.Length; i++)
00574                     {
00575                         RenderingThreads[i].Dispose();
00576                     }
00577                 }
00578
00579                 if (Contexts != null)
00580                 {
00581                     for (int i = 0; i < Contexts.Length; i++)
00582                     {
00583                         Contexts[i].Dispose();
00584                     }
00585                 }
00586             }
00587             disposedValue = true;
00588         }
00589     }
00590
00591 //<inheritDoc>
00592     public void Dispose()
```

```

00593     {
00594         Dispose(disposing: true);
00595         GC.SuppressFinalize(this);
00596     }
00597 }
00598 }
```

8.14 MuPDFOptionalContentGroupConfiguration.cs

```

00001 using System;
00002 using System.Collections.Generic;
00003 using System.Runtime.InteropServices;
00004 using System.Text;
00005
00006 namespace MuPDFCore
00007 {
00008 /// <summary>
00009 /// Contains information about optional content groups in a PDF document (also known as layers).
00010 /// </summary>
00011     public class MuPDFOptionalContentGroupData
00012     {
00013 /// <summary>
00014 /// The default optional content group configuration.
00015 /// </summary>
00016     public MuPDFOptionalContentGroupConfiguration DefaultConfiguration { get; }
00017
00018 /// <summary>
00019 /// Alternative optional content group configurations.
00020 /// </summary>
00021     public MuPDFOptionalContentGroupConfiguration[] AlternativeConfigurations { get; }
00022
00023 /// <summary>
00024 /// All optional content groups (layers) defined in this document.
00025 /// </summary>
00026     public MuPDFOptionalContentGroup[] OptionalContentGroups { get; }
00027     private MuPDFDocument OwnerDocument { get; }
00028
00029     internal static MuPDFOptionalContentGroupData Load(MuPDFDocument ownerDocument)
00030     {
00031         if (ownerDocument.NativePDFDocument != IntPtr.Zero)
00032         {
00033             MuPDFOptionalContentGroupData tbr = new MuPDFOptionalContentGroupData(ownerDocument);
00034
00035             if (tbr.DefaultConfiguration != null || tbr.AlternativeConfigurations.Length > 0 ||
00036                 tbr.OptionalContentGroups.Length > 0)
00037             {
00038                 return tbr;
00039             }
00040             else
00041             {
00042                 return null;
00043             }
00044         }
00045         else
00046         {
00047             return null;
00048         }
00049     }
00050
00051     private unsafe MuPDFOptionalContentGroupData(MuPDFDocument ownerDocument)
00052     {
00053         this.OwnerDocument = ownerDocument;
00054         this.DefaultConfiguration =
00055             MuPDFOptionalContentGroupConfiguration.GetDefaultConfiguration(OwnerDocument);
00056
00057         int alternativeOcgConfigs =
00058             NativeMethods.CountAlternativeOCGConfigs(this.OwnerDocument.OwnerContext.NativeContext,
00059             this.OwnerDocument.NativePDFDocument);
00060
00061         this.AlternativeConfigurations = new
00062             MuPDFOptionalContentGroupConfiguration[alternativeOcgConfigs];
00063
00064         for (int i = 0; i < alternativeOcgConfigs; i++)
00065         {
00066             this.AlternativeConfigurations[i] =
00067                 MuPDFOptionalContentGroupConfiguration.GetConfiguration(OwnerDocument, i);
00068         }
00069
00070         this.DefaultConfiguration?.Activate();
00071
00072         int ocgCount =
00073             NativeMethods.CountOptionalContentGroups(this.OwnerDocument.OwnerContext.NativeContext,
00074             this.OwnerDocument.NativePDFDocument);
```

```

00067
00068     this.OptionalContentGroups = new MuPDFOptionalContentGroup[ocgCount];
00069
00070     if (ocgCount > 0)
00071     {
00072         int[] ocgNameLengths = new int[ocgCount];
00073
00074         fixed (int* ocgNameLengthPtr = ocgNameLengths)
00075         {
00076
00077             NativeMethods.GetOptionalContentGroupNameLengths(this.OwnerDocument.OwnerContext.NativeContext,
00078                 this.OwnerDocument.NativePDFDocument, ocgCount, (IntPtr)ocgNameLengthPtr);
00079         }
00080
00081         byte[][] ocgNames = new byte[ocgCount][];
00082         GCHandle[] ocgNameHandles = new GCHandle[ocgCount];
00083         IntPtr[] ocgNamePointers = new IntPtr[ocgCount];
00084
00085         for (int i = 0; i < ocgCount; i++)
00086         {
00087             ocgNames[i] = new byte[ocgNameLengths[i]];
00088             ocgNameHandles[i] = GCHandle.Alloc(ocgNames[i], GCHandleType.Pinned);
00089             ocgNamePointers[i] = ocgNameHandles[i].AddrOfPinnedObject();
00090         }
00091
00092         fixed (IntPtr* ocgNamePointersPointer = ocgNamePointers)
00093         {
00094
00095             NativeMethods.GetOptionalContentGroups(this.OwnerDocument.OwnerContext.NativeContext,
00096                 this.OwnerDocument.NativePDFDocument, ocgCount, (IntPtr)ocgNamePointersPointer);
00097         }
00098
00099         for (int i = 0; i < ocgCount; i++)
00100         {
00101             ocgNameHandles[i].Free();
00102         }
00103
00104         this.OptionalContentGroups[i] = new MuPDFOptionalContentGroup(this.OwnerDocument,
00105             Encoding.UTF8.GetString(ocgNames[i]), i);
00106     }
00107
00108 /// <summary>
00109 /// Represents an optional content group configuration.
00110 /// </summary>
00111     public class MuPDFOptionalContentGroupConfiguration
00112     {
00113     /// <summary>
00114     /// The name of this optional content group configuration.
00115     /// </summary>
00116     public string Name { get; }
00117
00118     /// <summary>
00119     /// The creator of this optional content group configuration.
00120     /// </summary>
00121     public string Creator { get; }
00122
00123     /// <summary>
00124     /// Gets whether this is the default optional content group configuration for the document.
00125     /// </summary>
00126     public bool IsDefault { get => Index < 0; }
00127
00128     /// <summary>
00129     /// Optional content group "UI" elements associated with this configuration.
00130     /// </summary>
00131     public MuPDFOptionalContentGroupUIItem[] UI { get; }
00132
00133     private MuPDFDocument OwnerDocument { get; }
00134     private int Index { get; }
00135
00136     private unsafe MuPDFOptionalContentGroupConfiguration(string name, string creator, int index,
00137         MuPDFDocument ownerDocument)
00138     {
00139         this.Name = name;
00140         this.Creator = creator;
00141         this.Index = index;
00142         this.OwnerDocument = ownerDocument;
00143
00144         int uiCount = 0;
00145
00146         if (this.Name != null || this.Creator != null)
00147         {
00148             this.Activate();
00149         }
00150
00151         this.UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00152
00153         for (int i = 0; i < uiCount; i++)
00154         {
00155             this.UI[i] = new MuPDFOptionalContentGroupUIItem();
00156         }
00157
00158         this.UI[0].Name = this.Name;
00159         this.UI[0].Creator = this.Creator;
00160         this.UI[0].Index = this.Index;
00161
00162         this.UI[0].OwnerDocument = this.OwnerDocument;
00163
00164         this.UI[0].Activate();
00165
00166         this.UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00167
00168         for (int i = 0; i < uiCount; i++)
00169         {
00170             this.UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00171         }
00172
00173         this.UI[0].UI[0].Name = this.Name;
00174         this.UI[0].UI[0].Creator = this.Creator;
00175         this.UI[0].UI[0].Index = this.Index;
00176
00177         this.UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00178
00179         this.UI[0].UI[0].Activate();
00180
00181         this.UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00182
00183         for (int i = 0; i < uiCount; i++)
00184         {
00185             this.UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00186         }
00187
00188         this.UI[0].UI[0].UI[0].Name = this.Name;
00189         this.UI[0].UI[0].UI[0].Creator = this.Creator;
00190         this.UI[0].UI[0].UI[0].Index = this.Index;
00191
00192         this.UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00193
00194         this.UI[0].UI[0].UI[0].Activate();
00195
00196         this.UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00197
00198         for (int i = 0; i < uiCount; i++)
00199         {
00200             this.UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00201         }
00202
00203         this.UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00204         this.UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00205         this.UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00206
00207         this.UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00208
00209         this.UI[0].UI[0].UI[0].UI[0].Activate();
00210
00211         this.UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00212
00213         for (int i = 0; i < uiCount; i++)
00214         {
00215             this.UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00216         }
00217
00218         this.UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00219         this.UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00220         this.UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00221
00222         this.UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00223
00224         this.UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00225
00226         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00227
00228         for (int i = 0; i < uiCount; i++)
00229         {
00230             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00231         }
00232
00233         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00234         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00235         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00236
00237         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00238
00239         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00240
00241         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00242
00243         for (int i = 0; i < uiCount; i++)
00244         {
00245             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00246         }
00247
00248         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00249         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00250         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00251
00252         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00253
00254         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00255
00256         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00257
00258         for (int i = 0; i < uiCount; i++)
00259         {
00260             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00261         }
00262
00263         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00264         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00265         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00266
00267         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00268
00269         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00270
00271         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00272
00273         for (int i = 0; i < uiCount; i++)
00274         {
00275             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00276         }
00277
00278         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00279         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00280         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00281
00282         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00283
00284         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00285
00286         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00287
00288         for (int i = 0; i < uiCount; i++)
00289         {
00290             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00291         }
00292
00293         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00294         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00295         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00296
00297         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00298
00299         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00300
00301         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00302
00303         for (int i = 0; i < uiCount; i++)
00304         {
00305             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00306         }
00307
00308         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00309         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00310         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00311
00312         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00313
00314         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00315
00316         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00317
00318         for (int i = 0; i < uiCount; i++)
00319         {
00320             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00321         }
00322
00323         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00324         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00325         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00326
00327         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00328
00329         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00330
00331         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00332
00333         for (int i = 0; i < uiCount; i++)
00334         {
00335             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00336         }
00337
00338         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00339         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00340         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00341
00342         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00343
00344         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00345
00346         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00347
00348         for (int i = 0; i < uiCount; i++)
00349         {
00350             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00351         }
00352
00353         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00354         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00355         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00356
00357         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00358
00359         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00360
00361         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00362
00363         for (int i = 0; i < uiCount; i++)
00364         {
00365             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00366         }
00367
00368         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00369         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00370         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00371
00372         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00373
00374         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00375
00376         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00377
00378         for (int i = 0; i < uiCount; i++)
00379         {
00380             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00381         }
00382
00383         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00384         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00385         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00386
00387         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00388
00389         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00390
00391         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00392
00393         for (int i = 0; i < uiCount; i++)
00394         {
00395             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00396         }
00397
00398         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00399         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00400         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00401
00402         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00403
00404         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00405
00406         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00407
00408         for (int i = 0; i < uiCount; i++)
00409         {
00410             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00411         }
00412
00413         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00414         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00415         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00416
00417         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00418
00419         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00420
00421         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00422
00423         for (int i = 0; i < uiCount; i++)
00424         {
00425             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00426         }
00427
00428         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00429         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00430         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00431
00432         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00433
00434         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00435
00436         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00437
00438         for (int i = 0; i < uiCount; i++)
00439         {
00440             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00441         }
00442
00443         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00444         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00445         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00446
00447         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00448
00449         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00450
00451         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00452
00453         for (int i = 0; i < uiCount; i++)
00454         {
00455             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[i] = new MuPDFOptionalContentGroupUIItem();
00456         }
00457
00458         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Name = this.Name;
00459         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Creator = this.Creator;
00460         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Index = this.Index;
00461
00462         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].OwnerDocument = this.OwnerDocument;
00463
00464         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].Activate();
00465
00466         this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI = new MuPDFOptionalContentGroupUIItem[uiCount];
00467
00468         for (int i = 0; i < uiCount; i++)
00469         {
00470             this.UI[0].UI[0].UI[0].UI[0].UI[0].UI[0].UI
```

```

00148         uiCount =
00149             NativeMethods.CountOptionalContentGroupConfigUI(this.OwnerDocument.OwnerContext.NativeContext,
00150                 this.OwnerDocument.NativePDFDocument);
00151         }
00152         if (uiCount > 0)
00153         {
00154             int[] uiLabelLengths = new int[uiCount];
00155             fixed (int* uiLabelLengthsPtr = uiLabelLengths)
00156             {
00157
00158                 NativeMethods.ReadOptionalContentGroupUILabelLengths(this.OwnerDocument.OwnerContext.NativeContext,
00159                     this.OwnerDocument.NativePDFDocument, uiCount, (IntPtr)uiLabelLengthsPtr);
00160             }
00161             byte[][] labelBytes = new byte[uiCount][];
00162             GCHandle[] labelHandles = new GCHandle[uiCount];
00163             IntPtr[] labelPointers = new IntPtr[uiCount];
00164             for (int i = 0; i < uiCount; i++)
00165             {
00166                 labelBytes[i] = new byte[uiLabelLengths[i]];
00167                 labelHandles[i] = GCHandle.Alloc(labelBytes[i], GCHandleType.Pinned);
00168                 labelPointers[i] = labelHandles[i].AddrOfPinnedObject();
00169             }
00170             int[] depths = new int[uiCount];
00171             int[] types = new int[uiCount];
00172             int[] locked = new int[uiCount];
00173
00174             fixed (IntPtr* labelPointersPtr = labelPointers)
00175             fixed (int* depthsPtr = depths)
00176             fixed (int* typesPtr = types)
00177             fixed (int* lockedPtr = locked)
00178             {
00179
00180                 NativeMethods.ReadOptionalContentGroupUIs(this.OwnerDocument.OwnerContext.NativeContext,
00181                     this.OwnerDocument.NativePDFDocument, uiCount, (IntPtr)labelPointersPtr, (IntPtr)depthsPtr,
00182                     (IntPtr)typesPtr, (IntPtr)lockedPtr);
00183             }
00184             string[] labels = new string[uiCount];
00185             for (int i = 0; i < uiCount; i++)
00186             {
00187                 labels[i] = Encoding.UTF8.GetString(labelBytes[i]);
00188             }
00189             TemporaryUIItem superRootItem = new TemporaryUIItem(null, -1, -1, -1, -1, null);
00190             TemporaryUIItem rootItem = new TemporaryUIItem(null, -1, -1, 0, -1, superRootItem);
00191             TemporaryUIItem currItem = rootItem;
00192
00193             for (int i = 0; i < uiCount; i++)
00194             {
00195                 if (types[i] == 0) // Label
00196                 {
00197                     while (currItem.Depth >= depths[i])
00198                     {
00199                         currItem = currItem.Parent;
00200
00201                     TemporaryUIItem temp = new TemporaryUIItem(labels[i], types[i], locked[i],
00202                         depths[i], i, currItem);
00203                     currItem.Children.Add(temp);
00204                     currItem = temp;
00205                 }
00206                 else
00207                 {
00208                     while (((currItem.Depth == depths[i] && currItem.Type == 0) ||
00209                         (currItem.Depth == depths[i] - 1 && currItem.Type != 0)))
00210                     {
00211                         currItem = currItem.Parent;
00212
00213                     TemporaryUIItem temp = new TemporaryUIItem(labels[i], types[i], locked[i],
00214                         depths[i], i, currItem);
00215                     currItem.Children.Add(temp);
00216                     currItem = temp;
00217                 }
00218             for (int i = 0; i < superRootItem.Children.Count; i++)
00219             {
00220                 if (superRootItem.Children[i] != rootItem)
00221                 {
00222                     superRootItem.Children[i].Parent = rootItem;
00223                     rootItem.Children.Add(superRootItem.Children[i]);
00224                 }
00225             }
00226         }
00227     }
00228 }
```

```
00225         }
00226         rootItem.Parent = null;
00227
00228         this.UI = new MuPDFOptionalContentGroupUIItem[rootItem.Children.Count];
00229
00230         for (int i = 0; i < rootItem.Children.Count; i++)
00231         {
00232             this.UI[i] = rootItem.Children[i].Build(ownerDocument);
00233         }
00234     }
00235 }
00236 else
00237 {
00238     this.UI = new MuPDFOptionalContentGroupUIItem[0];
00239 }
00240 }
00241
00242     private class TemporaryUIItem
00243     {
00244         public string Label { get; set; }
00245         public int Type { get; set; }
00246         public int IsLocked { get; set; }
00247         public List<TemporaryUIItem> Children { get; set; }
00248         public int Depth { get; set; }
00249         public TemporaryUIItem Parent { get; set; }
00250         public int Index { get; set; }
00251
00252         public TemporaryUIItem(string label, int type, int isLocked, int depth, int index,
00253         TemporaryUIItem parent)
00254         {
00255             Label = label;
00256             Type = type;
00257             IsLocked = isLocked;
00258             Children = new List<TemporaryUIItem>();
00259             this.Depth = depth;
00260             this.Index = index;
00261             Parent = parent;
00262         }
00263
00264         public MuPDFOptionalContentGroupUIItem Build(MuPDFDocument ownerDocument)
00265         {
00266             MuPDFOptionalContentGroupUIItem[] children = new
00267             MuPDFOptionalContentGroupUIItem[this.Children.Count];
00268
00269             for (int i = 0; i < this.Children.Count; i++)
00270             {
00271                 children[i] = this.Children[i].Build(ownerDocument);
00272             }
00273
00274             switch (this.Type)
00275             {
00276                 case 0:
00277                     return new MuPDFOptionalContentGroupLabel(this.Label, children);
00278                 case 1:
00279                     return new MuPDFOptionalContentGroupCheckbox(this.Label, children,
00280                         this.IsLocked != 0, this.Index, ownerDocument);
00281                 case 2:
00282                     return new MuPDFOptionalContentGroupRadioButton(this.Label, children,
00283                         this.IsLocked != 0, this.Index, ownerDocument);
00284                 default:
00285                     throw new ArgumentException("Unknown UI item type: " + this.Type.ToString());
00286             }
00287         }
00288
00289         internal static unsafe MuPDFOptionalContentGroupConfiguration
00290         GetDefaultConfiguration(MuPDFDocument ownerDocument)
00291         {
00292             int nameLength = 0;
00293             int creatorLength = 0;
00294
00295             NativeMethods.ReadDefaultOCGConfigNameLength(ownerDocument.OwnerContext.NativeContext,
00296             ownerDocument.NativePDFDocument, ref nameLength, ref creatorLength);
00297
00298             if (nameLength > 0 || creatorLength > 0)
00299             {
00300                 byte[] configName = new byte[nameLength];
00301                 byte[] configCreator = new byte[creatorLength];
00302
00303                 fixed (byte* configNamePtr = configName)
00304                 fixed (byte* configCreatorPtr = configCreator)
00305                 {
00306                     NativeMethods.ReadDefaultOCGConfig(ownerDocument.OwnerContext.NativeContext,
00307                     ownerDocument.NativePDFDocument, nameLength, creatorLength, (IntPtr)configNamePtr,
00308                     (IntPtr)configCreatorPtr);
00309                 }
00310             }
00311         }
00312     }
00313 }
```

```

00304
00305         return new MuPDFOptionalContentGroupConfiguration(Encoding.UTF8.GetString(configName),
00306             Encoding.UTF8.GetString(configCreator), -1, ownerDocument);
00307     }
00308     {
00309         return null;
00310     }
00311 }
00312
00313     internal static unsafe MuPDFOptionalContentGroupConfiguration GetConfiguration(MuPDFDocument
00314     ownerDocument, int index)
00315     {
00316         int nameLength = 0;
00317         int creatorLength = 0;
00318
00319         NativeMethods.ReadOCGConfigNameLength(ownerDocument.OwnerContext.NativeContext,
00320             ownerDocument.NativePDFDocument, index, ref nameLength, ref creatorLength);
00321
00322         if (nameLength > 0 || creatorLength > 0)
00323         {
00324             byte[] configName = new byte[nameLength];
00325             byte[] configCreator = new byte[creatorLength];
00326
00327             fixed (byte* configNamePtr = configName)
00328             fixed (byte* configCreatorPtr = configCreator)
00329             {
00330                 NativeMethods.ReadOCGConfig(ownerDocument.OwnerContext.NativeContext,
00331                     ownerDocument.NativePDFDocument, index, nameLength, creatorLength, (IntPtr)configNamePtr,
00332                     (IntPtr)configCreatorPtr);
00333             }
00334         }
00335         return new MuPDFOptionalContentGroupConfiguration(Encoding.UTF8.GetString(configName),
00336             Encoding.UTF8.GetString(configCreator), index, ownerDocument);
00337     }
00338
00339 /// <summary>
00340 /// Activates this optional content group configuration.
00341 /// </summary>
00342     public void Activate()
00343     {
00344         if (this.IsEnabled)
00345         {
00346             NativeMethods.EnableDefaultOCGConfig(this.OwnerDocument.OwnerContext.NativeContext,
00347                 this.OwnerDocument.NativePDFDocument);
00348         }
00349         else
00350         {
00351             NativeMethods.EnableOCGConfig(this.OwnerDocument.OwnerContext.NativeContext,
00352                 this.OwnerDocument.NativePDFDocument, this.Index);
00353         }
00354     }
00355
00356 /// <summary>
00357 /// Represents an optional content group (also known as layer).
00358 /// </summary>
00359     public class MuPDFOptionalContentGroup
00360     {
00361         private int Index { get; }
00362         private MuPDFDocument OwnerDocument { get; }
00363
00364 /// <summary>
00365 /// The name of the optional content group.
00366 /// </summary>
00367         public string Name { get; }
00368
00369 /// <summary>
00370 /// Gets or sets whether the optional content group is enabled.
00371 /// </summary>
00372         public bool IsEnabled
00373         {
00374             get
00375             {
00376                 return
00377                     NativeMethods.GetOptionalContentGroupState(this.OwnerDocument.OwnerContext.NativeContext,
00378                     this.OwnerDocument.NativePDFDocument, this.Index) != 0;
00379             }
00380         }

```

```
NativeMethods.SetOptionalContentGroupState(this.OwnerDocument.OwnerContext.NativeContext,
    this.OwnerDocument.NativePDFDocument, this.Index, value ? 1 : 0);
00381     OwnerDocument.ClearCache();
00382 }
00383 }
00384
00385     internal MuPDFOptionalContentGroup(MuPDFDocument ownerDocument, string name, int index)
00386 {
00387     this.Name = name;
00388     this.Index = index;
00389     this.OwnerDocument = ownerDocument;
00390 }
00391 }
00392
00393 /// <summary>
00394 /// An optional content group UI element.
00395 /// </summary>
00396     public abstract class MuPDFOptionalContentGroupUIItem
00397 {
00398     /// <summary>
00399     /// The label for this optional content group UI element.
00400     /// </summary>
00401     public virtual string Label { get; }
00402
00403     /// <summary>
00404     /// Optional content group UI elements nested within this UI element (this may be empty, but it will
00405     /// not be <see langword="null"/>.
00406     /// </summary>
00407     public virtual MuPDFOptionalContentGroupUIItem[] Children { get; }
00408
00409     internal MuPDFOptionalContentGroupUIItem(string label, MuPDFOptionalContentGroupUIItem[]
children)
00410     {
00411         Label = label;
00412         Children = children;
00413     }
00414
00415     /// <summary>
00416     /// An optional content group UI element that can be enabled or disabled.
00417     /// </summary>
00418     public abstract class MuPDFOptionalContentGroupSelectableUIItem : MuPDFOptionalContentGroupUIItem
00419 {
00420     /// <summary>
00421     /// Gets or sets whether the optional content group UI element is enabled or not.
00422     /// </summary>
00423     public virtual bool IsEnabled
00424     {
00425         get =>
NativeMethods.ReadOptionalContentGroupUIState(this.OwnerDocument.OwnerContext.NativeContext,
    this.OwnerDocument.NativePDFDocument, this.Index) != 0;
00426         set
00427     {
00428         NativeMethods.SetOptionalContentGroupUIState(this.OwnerDocument.OwnerContext.NativeContext,
    this.OwnerDocument.NativeDocument, this.Index, value ? 1 : 0);
00429         OwnerDocument.ClearCache();
00430     }
00431 }
00432
00433     /// <summary>
00434     /// Indicates whether the state of the optional content group UI element is locked or can be changed.
00435     /// </summary>
00436     public virtual bool IsLocked { get; }
00437
00438     private int Index { get; }
00439
00440     private MuPDFDocument OwnerDocument { get; }
00441
00442     /// <summary>
00443     /// Toggle the state of the optional content group UI element.
00444     /// </summary>
00445     public virtual void Toggle()
00446     {
00447         NativeMethods.SetOptionalContentGroupUIState(this.OwnerDocument.OwnerContext.NativeContext,
    this.OwnerDocument.NativeDocument, this.Index, 2);
00448         OwnerDocument.ClearCache();
00449     }
00450
00451     internal MuPDFOptionalContentGroupSelectableUIItem(string label,
MuPDFOptionalContentGroupUIItem[] children, bool isLocked, int index, MuPDFDocument ownerDocument) :
base(label, children)
00452     {
00453         this.IsLocked = isLocked;
00454         this.Index = index;
00455         this.OwnerDocument = ownerDocument;
```

```

00456         }
00457     }
00458
00459 /// <summary>
00460 /// An optional content group UI element that should be represented as a check box.
00461 /// </summary>
00462     public class MuPDFOptionalContentGroupCheckbox : MuPDFOptionalContentGroupSelectableUIItem
00463     {
00464         internal MuPDFOptionalContentGroupCheckbox(string label, MuPDFOptionalContentGroupUIItem[]
00465             children, bool isLocked, int index, MuPDFDocument ownerDocument) : base(label, children, isLocked,
00466             index, ownerDocument) { }
00467     }
00468
00469 /// <summary>
00470 /// An optional content group UI element that should be represented as a radio button.
00471 /// </summary>
00472     public class MuPDFOptionalContentGroupRadioButton : MuPDFOptionalContentGroupSelectableUIItem
00473     {
00474         internal MuPDFOptionalContentGroupRadioButton(string label, MuPDFOptionalContentGroupUIItem[]
00475             children, bool isLocked, int index, MuPDFDocument ownerDocument) : base(label, children, isLocked,
00476             index, ownerDocument) { }
00477     }
00478
00479 /// <summary>
00480 /// An optional content group UI element that should be represented as a label.
00481 /// </summary>
00482     public class MuPDFOptionalContentGroupLabel : MuPDFOptionalContentGroupUIItem
00483     {
00484         internal MuPDFOptionalContentGroupLabel(string label, MuPDFOptionalContentGroupUIItem[]
00485             children) : base(label, children) { }
00486     }
00487 }
```

8.15 MuPDFOutline.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2024 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections;
00020 using System.Collections.Generic;
00021 using System.Linq;
00022 using System.Runtime.InteropServices;
00023 using System.Text;
00024
00025 namespace MuPDFCore
00026 {
00027     /// <summary>
00028     /// Represents a document outline (table of contents).
00029     /// </summary>
00030     public class MuPDFOutline : IReadOnlyList<MuPDFOutlineItem>
00031     {
00032         /// <summary>
00033         /// The outline items.
00034         /// </summary>
00035         public IReadOnlyList<MuPDFOutlineItem> Items { get; }
00036
00037         internal MuPDFDocument OwnerDocument { get; }
00038
00039         /// <summary>
00040         /// Load the outline for a document.
00041         /// </summary>
00042         /// <param name="context">A <see cref="MuPDFContext"/> to store resources and the exception
00043         /// stack.</param>
00044         /// <param name="document">The document whose outline should be loaded.</param>
00045         internal MuPDFOutline(MuPDFContext context, MuPDFDocument document)
00046         {
00047             this.OwnerDocument = document;
00048             IntPtr outline = NativeMethods.LoadOutline(context.NativeContext,
00049                 document.NativeDocument);
```

```
00048         if (outline != IntPtr.Zero)
00049     {
00050         this.Items = MuPDFOutlineItem.TraverseOutline(outline, this).ToArray();
00051         NativeMethods.DisposeOutline(context.NativeContext, outline);
00052     }
00053     else
00054     {
00055         this.Items = new MuPDFOutlineItem[0];
00056     }
00057 }
00058 }
00059 }
00060
00061 /// <inheritdoc />
00062     public int Count => Items.Count;
00063
00064 /// <inheritdoc />
00065     public MuPDFOutlineItem this[int index] => Items[index];
00066
00067 /// <inheritdoc />
00068     public IEnumarator<MuPDFOutlineItem> GetEnumerator()
00069     {
00070         return Items.GetEnumerator();
00071     }
00072
00073 /// <inheritdoc />
00074     IEnumerator IEnumerable.GetEnumerator()
00075     {
00076         return ((IEnumerable)Items).GetEnumerator();
00077     }
00078 }
00079
00080 /// <summary>
00081 /// Represents an item in a document outline (table of contents).
00082 /// </summary>
00083     public class MuPDFOutlineItem
00084     {
00085         /// <summary>
00086         /// The title of the item. This can be null if the item does not have any text.
00087         /// </summary>
00088         public string Title { get; }
00089
00090         /// <summary>
00091         /// The destination in the document to be displayed when this outline item is activated. This can be
00092         /// null if the item has no destination.
00093         public string Uri { get; }
00094
00095         /// <summary>
00096         /// Locations within the document are referred to in terms of chapter and page, rather than just a
00097         /// page number. For some documents (such as epub documents with large numbers of pages broken into many
00098         /// chapters) this can make navigation much faster as only the required chapter needs to be decoded at a
00099         /// time.
00100         /// </summary>
00101         public int Chapter { get; }
00102
00103         /// <summary>
00104         /// The page number of an internal link, relative to the specified <see cref="Chapter"/>, or -1 for
00105         /// external links or links with no destination.
00106         public int Page { get; }
00107
00108         private int? CachedPageNumber = null;
00109
00110         /// <summary>
00111         /// The overall page number of an internal link. This is determined on first access, and it might
00112         /// cause a large number of chapters to be laid out to determine it.
00113         public int PageNumber
00114         {
00115             get
00116             {
00117                 if (CachedPageNumber == null)
00118                 {
00119                     CachedPageNumber =
00120                         NativeMethods.GetPageNumber(this.OwnerOutline.OwnerDocument.OwnerContext.NativeContext,
00121                         this.OwnerOutline.OwnerDocument.NativeDocument, this.Chapter, this.Page);
00122                 }
00123             }
00124
00125             return CachedPageNumber.Value;
00126         }
00127     }
00128
00129     /// <summary>
00130     /// The location on the page of the item pointed to by this outline item.
00131     /// </summary>
00132     public PointF Location { get; }
```

```

00127
00128 /// <summary>
00129 /// The sub items of this outline item (may be empty, but will not be null).
00130 /// </summary>
00131     public IReadOnlyList<MuPDFOutlineItem> Children { get; private set; }
00132
00133     internal MuPDFOutline OwnerOutline { get; }
00134
00135     [StructLayout(LayoutKind.Sequential)]
00136     private struct fz_outline
00137     {
00138         public int refs;
00139         public IntPtr title;
00140         public IntPtr uri;
00141         public int chapter;
00142         public int page;
00143         public float x;
00144         public float y;
00145         public IntPtr next;
00146         public IntPtr down;
00147         public int is_open;
00148     }
00149
00150 /// <summary>
00151 /// Get the length of a null-terminated C string.
00152 /// </summary>
00153 /// <param name="stringAddress">A pointer to the string.</param>
00154 /// <returns>The length of the string in bytes.</returns>
00155     private static unsafe int strlen(byte* stringAddress)
00156     {
00157         byte* originalAddress = stringAddress;
00158
00159         while (*stringAddress != 0)
00160         {
00161             stringAddress++;
00162         }
00163
00164         return (int)(stringAddress - originalAddress);
00165     }
00166
00167 /// <summary>
00168 /// Convert a null-terminated C string in UTF8 encoding to a .NET <see langword="string"/>.
00169 /// </summary>
00170 /// <param name="stringAddress">A pointer to the string.</param>
00171 /// <returns>A .NET <see langword="string"/>.</returns>
00172     private static unsafe string PtrToStringUTF8(byte* stringAddress) =>
00173         Encoding.UTF8.GetString(stringAddress, strlen(stringAddress));
00174
00175 /// <summary>
00176 /// Create a new <see cref="MuPDFOutlineItem"/> from a native pointer.
00177 /// </summary>
00178 /// <param name="nativeOutlineItem">The pointer to the native outline item.</param>
00179 /// <param name="next">A pointer to the next item in the outline. This will be <see
00180 /// <param name="ownerOutline">The <see cref="MuPDFOutline"/> that contains the current item.</param>
00181     private unsafe MuPDFOutlineItem(IntPtr nativeOutlineItem, out IntPtr next, MuPDFOutline
00182         ownerOutline)
00183     {
00184         fz_outline nativeItem = Marshal.PtrToStructure<fz_outline>(nativeOutlineItem);
00185
00186         string title = nativeItem.title == IntPtr.Zero ? null :
00187             PtrToStringUTF8((byte*)nativeItem.title);
00188         string uri = nativeItem.uri == IntPtr.Zero ? null :
00189             PtrToStringUTF8((byte*)nativeItem.uri);
00190
00191         this.Title = title;
00192         this.Uri = uri;
00193         this.Chapter = nativeItem.chapter;
00194         this.Page = nativeItem.page;
00195         this.Location = new PointF(nativeItem.x, nativeItem.y);
00196         this.OwnerOutline = ownerOutline;
00197
00198         next = nativeItem.next;
00199
00200         if (nativeItem.down != IntPtr.Zero)
00201         {
00202             this.Children = TraverseOutline(nativeItem.down, ownerOutline).ToArray();
00203         }
00204     }
00205
00206 /// <summary>
00207 /// Recursively traverse the outline, loading information about all the items.
00208 /// </summary>

```

```

00209 /// <param name="firstOutlineItem">The first item in the outline.</param>
00210 /// <param name="ownerOutline">The <see cref="MuPDFOutline"/> that contains the current item.</param>
00211 /// <returns>A collection of <see cref="MuPDFOutlineItem"/> containing all the information in the
00212     outline.</returns>
00213     internal static IEnumerable<MuPDFOutlineItem> TraverseOutline(IntPtr firstOutlineItem,
00214     MuPDFOutline ownerOutline)
00215     {
00216         IntPtr currOutlineItem = firstOutlineItem;
00217         while (currOutlineItem != IntPtr.Zero)
00218         {
00219             MuPDFOutlineItem item = new MuPDFOutlineItem(currOutlineItem, out IntPtr next,
00220             ownerOutline);
00221             yield return item;
00222             currOutlineItem = next;
00223         }
00224     }
00225 }
```

8.16 MuPDFPage.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019 using System.Collections;
00020 using System.Collections.Generic;
00021 using System.Xml.Linq;
00022
00023 namespace MuPDFCore
00024 {
00025 /// <summary>
00026 /// A wrapper over a MuPDF page object, which contains information about the page's boundaries.
00027 /// </summary>
00028     public class MuPDFPage : IDisposable
00029     {
00030     /// <summary>
00031     /// The page's bounds at 72 DPI. Read-only.
00032     /// </summary>
00033         public Rectangle Bounds { get; }
00034
00035     /// <summary>
00036     /// The number of this page in the original document.
00037     /// </summary>
00038         public int PageNumber { get; }
00039
00040         private MuPDFLinks CachedLinks = null;
00041
00042     /// <summary>
00043     /// The links contained within the <see cref="MuPDFPage"/>. This collection is populated on first
00044     /// access.
00045     public MuPDFLinks Links
00046     {
00047         get
00048         {
00049             if (CachedLinks == null)
00050             {
00051                 CachedLinks = new MuPDFLinks(this);
00052             }
00053
00054             return CachedLinks;
00055         }
00056     }
00057
00058     /// <summary>
00059     /// A pointer to the native page object.

```

```

00060 /// </summary>
00061     internal readonly IntPtr NativePage;
00062
00063 /// <summary>
00064 /// The context that owns the document from which this page was extracted.
00065 /// </summary>
00066     private readonly MuPDFContext OwnerContext;
00067
00068 /// <summary>
00069 /// The document from which the page was extracted.
00070 /// </summary>
00071     internal readonly MuPDFDocument OwnerDocument;
00072
00073 /// <summary>
00074 /// The page's original bounds. Read-only.
00075 /// </summary>
00076     internal Rectangle OriginalBounds { get; }
00077
00078 /// <summary>
00079 /// Create a new <see cref="MuPDFPage"/> object from the specified document.
00080 /// </summary>
00081 /// <param name="context">The context that owns the document.</param>
00082 /// <param name="document">The document from which the page should be extracted.</param>
00083 /// <param name="number">The number of the page that should be extracted (starting at 0).</param>
00084     internal MuPDFPage(MuPDFContext context, MuPDFDocument document, int number)
00085     {
00086         if (document.EncryptionState == EncryptionState.Encrypted)
00087         {
00088             throw new DocumentLockedException("A password is necessary to render the document!");
00089         }
00090
00091         this.OwnerContext = context;
00092         this.OwnerDocument = document;
00093         this.PageNumber = number;
00094
00095         float x = 0;
00096         float y = 0;
00097         float w = 0;
00098         float h = 0;
00099
00100         ExitCodes result = (ExitCodes)NativeMethods.LoadPage(context.NativeContext,
00101             document.NativeDocument, number, ref NativePage, ref x, ref y, ref w, ref h);
00102
00103         double sX = Math.Round(x * document.ImageXRes / 72.0 * 1000) / 1000;
00104         double sY = Math.Round(y * document.ImageYRes / 72.0 * 1000) / 1000;
00105         double sW = Math.Round(w * document.ImageXRes / 72.0 * 1000) / 1000;
00106         double sH = Math.Round(h * document.ImageYRes / 72.0 * 1000) / 1000;
00107
00108         this.Bounds = new Rectangle(sX, sY, sX + sW, sY + sH);
00109         this.OriginalBounds = new Rectangle(x, y, x + w, y + h);
00110
00111         switch (result)
00112         {
00113             case ExitCodes.EXIT_SUCCESS:
00114                 break;
00115             case ExitCodes.ERR_CANNOT_LOAD_PAGE:
00116                 throw new MuPDFException("Cannot load page", result);
00117             case ExitCodes.ERR_CANNOT_COMPUTE_BOUNDS:
00118                 throw new MuPDFException("Cannot compute bounds", result);
00119             default:
00120                 throw new MuPDFException("Unknown error", result);
00121         }
00122     }
00123 /// <summary>
00124 /// Gets the specified bounding box from the page.
00125 /// </summary>
00126 /// <param name="boxType">The type of bounding box to get.</param>
00127 /// <param name="rescale">If this is <see langword="true"/>, the bounding box is rescaled so that it
00128 /// is expressed in the same resolution units as the underlying document. If this is <see
00129 /// langword="false"/>, the raw value is returned (at 72 dpi).</param>
00130 /// <returns>A <see cref="Rectangle"/> corresponding to the specified bounding box.</returns>
00131 /// <exception cref="MuPDFException">Thrown if the bounding box cannot be computed or if another error
00132 /// occurs.</exception>
00133     public Rectangle GetBoundingBox(BoxType boxType, bool rescale = true)
00134     {
00135         float x = 0;
00136         float y = 0;
00137         float w = 0;
00138         float h = 0;
00139
00140         ExitCodes result = (ExitCodes)NativeMethods.GetPageBox(this.OwnerContext.NativeContext,
00141             this.NativePage, (int)boxType, ref x, ref y, ref w, ref h);
00142
00143         switch (result)
00144         {
00145             case ExitCodes.EXIT_SUCCESS:

```

```
00142             break;
00143         case ExitCodes.ERR_CANNOT_COMPUTE_BOUNDS:
00144             throw new MuPDFException("Cannot compute bounds", result);
00145         default:
00146             throw new MuPDFException("Unknown error", result);
00147     }
00148 
00149     if (rescale)
00150     {
00151         double sX = Math.Round(x * this.OwnerDocument.ImageXRes / 72.0 * 1000) / 1000;
00152         double sY = Math.Round(y * this.OwnerDocument.ImageYRes / 72.0 * 1000) / 1000;
00153         double sW = Math.Round(w * this.OwnerDocument.ImageXRes / 72.0 * 1000) / 1000;
00154         double sH = Math.Round(h * this.OwnerDocument.ImageYRes / 72.0 * 1000) / 1000;
00155 
00156         return new Rectangle(sX, sY, sX + sW, sY + sH);
00157     }
00158     else
00159     {
00160         return new Rectangle(x, y, x + w, y + h);
00161     }
00162 }
00163 private bool disposedValue;
00164 
00165 ///<inheritdoc>
00166 protected virtual void Dispose(bool disposing)
00167 {
00168     if (!disposedValue)
00169     {
00170         if (OwnerContext.disposedValue)
00171         {
00172             throw new LifetimeManagementException<MuPDFPage, MuPDFContext>(this, OwnerContext,
00173             this.NativePage, OwnerContext.NativeContext);
00174         }
00175         if (disposing)
00176         {
00177             this.Links.Dispose();
00178         }
00179         NativeMethods.DisposePage(OwnerContext.NativeContext, NativePage);
00180         disposedValue = true;
00181     }
00182 }
00183 
00184 ///<inheritdoc>
00185 ~MuPDFPage()
00186 {
00187     if (NativePage != IntPtr.Zero)
00188     {
00189         Dispose(disposing: false);
00190     }
00191 }
00192 
00193 ///<inheritdoc>
00194 public void Dispose()
00195 {
00196     Dispose(disposing: true);
00197     GC.SuppressFinalize(this);
00198 }
00199 }
00200 
00201 /// <summary>
00202 /// A lazy collection of <see cref="MuPDFPage"/>s. Each page is loaded from the document as it is
00203 /// requested for the first time.
00204 /// </summary>
00205 public class MuPDFPageCollection : IReadOnlyList<MuPDFPage>, IDisposable
00206 {
00207     /// <summary>
00208     /// The internal store of the pages.
00209     private readonly MuPDFPage[] Pages;
00210 
00211     /// <summary>
00212     /// The context that owns the document from which the pages were extracted.
00213     /// </summary>
00214     private readonly MuPDFContext OwnerContext;
00215 
00216     /// <summary>
00217     /// The document from which the pages were extracted.
00218     /// </summary>
00219     private readonly MuPDFDocument OwnerDocument;
00220 
00221     /// <summary>
00222     /// The number of pages in the collection.
00223     /// </summary>
00224     public int Length { get { return Pages.Length; } }
00225 
00226     /// <summary>
```

```
00227 /// The number of pages in the collection.
00228 /// </summary>
00229     public int Count { get { return Pages.Length; } }
00230
00231 /// <summary>
00232 /// Get a page from the collection.
00233 /// </summary>
00234 /// <param name="index">The number of the page (starting at 0).</param>
00235 /// <returns>The specified <see cref="MuPDFPage"/>.</returns>
00236     public MuPDFPage this[int index]
00237     {
00238         get
00239         {
00240             if (index < 0 || index > Pages.Length)
00241             {
00242                 throw new IndexOutOfRangeException();
00243             }
00244
00245             if (Pages[index] == null)
00246             {
00247                 Pages[index] = new MuPDFPage(OwnerContext, OwnerDocument, index);
00248             }
00249
00250             return Pages[index];
00251         }
00252     }
00253
00254 /// <summary>
00255 /// Create a new <see cref="MuPDFPageCollection"/> from the specified document, containing the
00256 /// specified number of pages.
00257 /// </summary>
00258 /// <param name="context">The context that owns the document.</param>
00259 /// <param name="document">The document from which the pages should be extracted.</param>
00260     internal MuPDFPageCollection(MuPDFContext context, MuPDFDocument document, int length)
00261     {
00262         Pages = new MuPDFPage[length];
00263         OwnerContext = context;
00264         OwnerDocument = document;
00265     }
00266
00267 ///<inheritdoc/>
00268     public IEnumerator<MuPDFPage> GetEnumerator()
00269     {
00270         for (int i = 0; i < Pages.Length; i++)
00271         {
00272             if (Pages[i] == null)
00273             {
00274                 Pages[i] = new MuPDFPage(OwnerContext, OwnerDocument, i);
00275             }
00276         }
00277
00278         return ((IEnumerable<MuPDFPage>)Pages).GetEnumerator();
00279     }
00280
00281 ///<inheritdoc/>
00282     IEnumerator IEnumerable.GetEnumerator()
00283     {
00284         return GetEnumerator();
00285     }
00286
00287     private bool disposedValue;
00288
00289 ///<inheritdoc/>
00290     protected virtual void Dispose(bool disposing)
00291     {
00292         if (!disposedValue)
00293         {
00294             if (disposing)
00295             {
00296                 for (int i = 0; i < Pages.Length; i++)
00297                 {
00298                     Pages[i]?.Dispose();
00299                 }
00300             }
00301
00302             disposedValue = true;
00303         }
00304     }
00305
00306 ///<inheritdoc/>
00307     public void Dispose()
00308     {
00309         Dispose(disposing: true);
00310         GC.SuppressFinalize(this);
00311     }
00312 }
```

```
00313 }
```

8.17 Rectangles.cs

```
00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019
00020 namespace MuPDFCore
00021 {
00022 /// <summary>
00023 /// Represents the size of a rectangle.
00024 /// </summary>
00025     public struct Size
00026     {
00027         /// <summary>
00028         /// The width of the rectangle.
00029         /// </summary>
00030         public float Width;
00031
00032         /// <summary>
00033         /// The height of the rectangle.
00034         /// </summary>
00035         public float Height;
00036
00037         /// <summary>
00038         /// Create a new <see cref="Size"/> with the specified width and height.
00039         /// </summary>
00040         /// <param name="width">The width of the rectangle.</param>
00041         /// <param name="height">The height of the rectangle.</param>
00042         public Size(float width, float height)
00043         {
00044             Width = width;
00045             Height = height;
00046         }
00047
00048         /// <summary>
00049         /// Create a new <see cref="Size"/> with the specified width and height.
00050         /// </summary>
00051         /// <param name="width">The width of the rectangle.</param>
00052         /// <param name="height">The height of the rectangle.</param>
00053         public Size(double width, double height)
00054         {
00055             Width = (float)width;
00056             Height = (float)height;
00057         }
00058
00059         /// <summary>
00060         /// Split the size into the specified number of <see cref="Rectangle"/>s.
00061         /// </summary>
00062         /// <param name="divisions">The number of rectangles in which the size should be split. This must be
00063         /// factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <paramref
00064         /// name="divisions"/> that satisfies this condition is used.</param>
00065         /// <returns>An array of <see cref="Rectangle"/>s that when positioned properly cover an area of the
00066         /// size of this object.</returns>
00067         public Rectangle[] Split(int divisions)
00068         {
00069             divisions = Utils.GetAcceptableNumber(divisions);
00070
00071             Rectangle[] tbr = new Rectangle[divisions];
00072
00073             bool isVertical = this.Height > this.Width;
00074
00075             if (divisions == 1)
00076             {
00077                 tbr[0] = new Rectangle(0, 0, Width, Height);
00078             }
00079             else if (divisions == 2)
```



```

00163                     tbr[pos] = new Rectangle(largerDivisions[i].X0 + currDivision[j].X0,
00164                                         largerDivisions[i].Y0 + currDivision[j].Y0, largerDivisions[i].X0 + currDivision[j].X1,
00165                                         largerDivisions[i].Y0 + currDivision[j].Y1);
00166                     pos++;
00167                 }
00168             break;
00169         }
00170     }
00171 }
00172 return tbr;
00173 }
00174 }
00175 }
00176 }
00177
00178 /// <summary>
00179 /// Represents the size of a rectangle using only integer numbers.
00180 /// </summary>
00181 public struct RoundedSize
00182 {
00183     /// <summary>
00184     /// The width of the rectangle.
00185     /// </summary>
00186     public int Width;
00187
00188     /// <summary>
00189     /// The height of the rectangle.
00190     /// </summary>
00191     public int Height;
00192
00193     /// <summary>
00194     /// Create a new <see cref="RoundedSize"/> with the specified width and height.
00195     /// </summary>
00196     /// <param name="width">The width of the rectangle.</param>
00197     /// <param name="height">The height of the rectangle.</param>
00198     public RoundedSize(int width, int height)
00199     {
00200         Width = width;
00201         Height = height;
00202     }
00203
00204     /// <summary>
00205     /// Split the size into the specified number of <see cref="RoundedRectangle"/>s.
00206     /// </summary>
00207     /// <param name="divisions">The number of rectangles in which the size should be split. This must be
00208     /// factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than <paramref
00209     /// name="divisions"/> that satisfies this condition is used.</param>
00210     /// <returns>An array of <see cref="RoundedRectangle"/>s that when positioned properly cover an area
00211     /// of the size of this object.</returns>
00212     public RoundedRectangle[] Split(int divisions)
00213     {
00214         divisions = Utils.GetAcceptableNumber(divisions);
00215         RoundedRectangle[] tbr = new RoundedRectangle[divisions];
00216
00217         bool isVertical = this.Height > this.Width;
00218
00219         if (divisions == 1)
00220         {
00221             tbr[0] = new RoundedRectangle(0, 0, Width, Height);
00222         }
00223         else if (divisions == 2)
00224         {
00225             if (isVertical)
00226             {
00227                 tbr[0] = new RoundedRectangle(0, 0, Width, Height / 2);
00228                 tbr[1] = new RoundedRectangle(0, Height / 2, Width, Height);
00229             }
00230             else
00231             {
00232                 tbr[0] = new RoundedRectangle(0, 0, Width / 2, Height);
00233                 tbr[1] = new RoundedRectangle(Width / 2, 0, Width, Height);
00234             }
00235         }
00236         else if (divisions == 3)
00237         {
00238             if (isVertical)
00239             {
00240                 tbr[0] = new RoundedRectangle(0, 0, Width / 2, 2 * Height / 3);
00241                 tbr[1] = new RoundedRectangle(Width / 2, 0, Width, 2 * Height / 3);
00242                 tbr[2] = new RoundedRectangle(0, 2 * Height / 3, Width, Height);
00243             }
00244             else
00245             {
00246                 tbr[0] = new RoundedRectangle(0, 0, 2 * Width / 3, Height / 2);
00247             }
00248         }
00249     }

```

```

00245             tbr[1] = new RoundedRectangle(0, Height / 2, 2 * Width / 3, Height);
00246             tbr[2] = new RoundedRectangle(2 * Width / 3, 0, Width, Height);
00247         }
00248     }
00249     else if (divisions == 5)
00250     {
00251         if (isVertical)
00252         {
00253             tbr[0] = new RoundedRectangle(0, 0, Width / 2, 2 * Height / 5);
00254             tbr[1] = new RoundedRectangle(Width / 2, 0, Width, 2 * Height / 5);
00255             tbr[2] = new RoundedRectangle(0, 2 * Height / 5, Width / 2, 4 * Height / 5);
00256             tbr[3] = new RoundedRectangle(Width / 2, 2 * Height / 5, Width, 4 * Height / 5);
00257             tbr[4] = new RoundedRectangle(0, 4 * Height / 5, Width, Height);
00258         }
00259     else
00260     {
00261         tbr[0] = new RoundedRectangle(0, 0, 2 * Width / 5, Height / 2);
00262         tbr[1] = new RoundedRectangle(0, Height / 2, 2 * Width / 5, Height);
00263         tbr[2] = new RoundedRectangle(2 * Width / 5, 0, 4 * Width / 5, Height / 2);
00264         tbr[3] = new RoundedRectangle(2 * Width / 5, Height / 2, 4 * Width / 5, Height);
00265         tbr[4] = new RoundedRectangle(4 * Width / 5, 0, Width, Height);
00266     }
00267 }
00268 else if (divisions == 7)
00269 {
00270     if (isVertical)
00271     {
00272         tbr[0] = new RoundedRectangle(0, 0, Width / 2, 2 * Height / 7);
00273         tbr[1] = new RoundedRectangle(Width / 2, 0, Width, 2 * Height / 7);
00274         tbr[2] = new RoundedRectangle(0, 2 * Height / 7, Width / 2, 4 * Height / 7);
00275         tbr[3] = new RoundedRectangle(Width / 2, 2 * Height / 7, Width, 4 * Height / 7);
00276         tbr[4] = new RoundedRectangle(0, 4 * Height / 7, Width / 2, 6 * Height / 7);
00277         tbr[5] = new RoundedRectangle(Width / 2, 4 * Height / 7, Width, 6 * Height / 7);
00278         tbr[6] = new RoundedRectangle(0, 6 * Height / 7, Width, Height);
00279     }
00280 else
00281 {
00282     tbr[0] = new RoundedRectangle(0, 0, 2 * Width / 7, Height / 2);
00283     tbr[1] = new RoundedRectangle(0, Height / 2, 2 * Width / 7, Height);
00284     tbr[2] = new RoundedRectangle(2 * Width / 7, 0, 4 * Width / 7, Height / 2);
00285     tbr[3] = new RoundedRectangle(2 * Width / 7, Height / 2, 4 * Width / 7, Height);
00286     tbr[4] = new RoundedRectangle(4 * Width / 7, 0, 6 * Width / 7, Height / 2);
00287     tbr[5] = new RoundedRectangle(4 * Width / 7, Height / 2, 6 * Width / 7, Height);
00288     tbr[6] = new RoundedRectangle(6 * Width / 7, 0, Width, Height);
00289 }
00290 else
00291 {
00292     for (int divisorInd = 0; divisorInd < Utils.AcceptableDivisors.Length; divisorInd++)
00293     {
00294         if (divisions % Utils.AcceptableDivisors[divisorInd] == 0)
00295         {
00296             RoundedRectangle[] largerDivisions = this.Split(divisions /
00297                 Utils.AcceptableDivisors[divisorInd]);
00298             int pos = 0;
00299
00300             for (int i = 0; i < largerDivisions.Length; i++)
00301             {
00302                 RoundedRectangle s = new RoundedRectangle(largerDivisions[i].Width,
00303                     largerDivisions[i].Height);
00304                 RoundedRectangle[] currDivision =
00305                     s.Split(Utils.AcceptableDivisors[divisorInd]);
00306
00307                 for (int j = 0; j < currDivision.Length; j++)
00308                 {
00309                     tbr[pos] = new RoundedRectangle(largerDivisions[i].X0 +
00310                         currDivision[j].X0, largerDivisions[i].Y0 + currDivision[j].Y0, largerDivisions[i].X0 +
00311                         currDivision[j].X1, largerDivisions[i].Y0 + currDivision[j].Y1);
00312                     pos++;
00313                 }
00314             }
00315         }
00316     }
00317
00318     return tbr;
00319 }
00320 }
00321 }
00322
00323 /// <summary>
00324 /// Represents a rectangle.
00325 /// </summary>
00326 public struct Rectangle

```

```
00327      {
00328  /// <summary>
00329  /// The left coordinate of the rectangle.
00330  /// </summary>
00331      public float X0;
00332
00333  /// <summary>
00334  /// The top coordinate of the rectangle.
00335  /// </summary>
00336      public float Y0;
00337
00338  /// <summary>
00339  /// The right coordinate of the rectangle.
00340  /// </summary>
00341      public float X1;
00342
00343  /// <summary>
00344  /// The bottom coordinate of the rectangle.
00345  /// </summary>
00346      public float Y1;
00347
00348  /// <summary>
00349  /// The width of the rectangle.
00350  /// </summary>
00351      public float Width => X1 - X0;
00352
00353  /// <summary>
00354  /// The height of the rectangle.
00355  /// </summary>
00356      public float Height => Y1 - Y0;
00357
00358  /// <summary>
00359  /// Create a new <see cref="Rectangle"/> from the specified coordinates.
00360  /// </summary>
00361  /// <param name="x0">The left coordinate of the rectangle.</param>
00362  /// <param name="y0">The top coordinate of the rectangle.</param>
00363  /// <param name="x1">The right coordinate of the rectangle.</param>
00364  /// <param name="y1">The bottom coordinate of the rectangle.</param>
00365      public Rectangle(float x0, float y0, float x1, float y1)
00366      {
00367          X0 = x0;
00368          Y0 = y0;
00369          X1 = x1;
00370          Y1 = y1;
00371      }
00372
00373  /// <summary>
00374  /// Create a new <see cref="Rectangle"/> from the specified coordinates.
00375  /// </summary>
00376  /// <param name="x0">The left coordinate of the rectangle.</param>
00377  /// <param name="y0">The top coordinate of the rectangle.</param>
00378  /// <param name="x1">The right coordinate of the rectangle.</param>
00379  /// <param name="y1">The bottom coordinate of the rectangle.</param>
00380      public Rectangle(double x0, double y0, double x1, double y1)
00381      {
00382          X0 = (float)x0;
00383          Y0 = (float)y0;
00384          X1 = (float)x1;
00385          Y1 = (float)y1;
00386      }
00387
00388  /// <summary>
00389  /// Round the rectangle's coordinates to the closest integers.
00390  /// </summary>
00391  /// <returns>A <see cref="RoundedRectangle"/> with the rounded coordinates.</returns>
00392      public RoundedRectangle Round()
00393      {
00394          return new RoundedRectangle(
00395              (int)Math.Floor(X0 + 0.001),
00396              (int)Math.Floor(Y0 + 0.001),
00397              (int)Math.Ceiling(X1 - 0.001),
00398              (int)Math.Ceiling(Y1 - 0.001)
00399          );
00400      }
00401
00402  /// <summary>
00403  /// Round the rectangle's coordinates to the closest integers, applying the specified zoom factor.
00404  /// </summary>
00405  /// <param name="zoom">The zoom factor to apply.</param>
00406  /// <returns>A <see cref="RoundedRectangle"/> with the rounded coordinates.</returns>
00407      public RoundedRectangle Round(double zoom)
00408      {
00409          return new RoundedRectangle(
00410              (int)Math.Floor(X0 * (float)zoom + 0.001),
00411              (int)Math.Floor(Y0 * (float)zoom + 0.001),
00412              (int)Math.Ceiling(X1 * (float)zoom - 0.001),
00413              (int)Math.Ceiling(Y1 * (float)zoom - 0.001)
```

```

00414         );
00415     }
00416
00417 /// <summary>
00418 /// Split the rectangle into the specified number of <see cref="Rectangle"/>s.
00419 /// </summary>
00420 /// <param name="divisions">The number of rectangles in which the rectangle should be split. This
00421 must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than
00422 <paramref name="divisions"/> that satisfies this condition is used.</param>
00423 /// <returns>An array of <see cref="Rectangle"/>s that when positioned properly cover the same area as
00424 this object.</returns>
00425     public Rectangle[] Split(int divisions)
00426     {
00427         Size s = new Size(this.Width, this.Height);
00428
00429         Rectangle[] splitSize = s.Split(divisions);
00430
00431         Rectangle[] tbr = new Rectangle[divisions];
00432
00433         for (int i = 0; i < splitSize.Length; i++)
00434         {
00435             tbr[i] = new Rectangle(this.X0 + splitSize[i].X0, this.Y0 + splitSize[i].Y0, this.X0 +
00436             splitSize[i].X1, this.Y0 + splitSize[i].Y1);
00437         }
00438
00439     return tbr;
00440 }
00441
00442 /// <summary>
00443 /// Compute the intersection between this <see cref="Rectangle"/> and another one.
00444 /// </summary>
00445 /// <param name="other">The other <see cref="Rectangle"/> to intersect with this instance.</param>
00446 /// <returns>The intersection between the two <see cref="Rectangle"/>s.</returns>
00447     public Rectangle Intersect(Rectangle other)
00448     {
00449         float x0 = Math.Max(this.X0, other.X0);
00450         float y0 = Math.Max(this.Y0, other.Y0);
00451
00452         float x1 = Math.Min(this.X1, other.X1);
00453         float y1 = Math.Min(this.Y1, other.Y1);
00454
00455         if (x1 <= x0 || y1 <= y0)
00456         {
00457             return new Rectangle(0, 0, 0, 0);
00458         }
00459     }
00460
00461 /// <summary>
00462 /// Checks whether this <see cref="Rectangle"/> contains another <see cref="Rectangle"/>.
00463 /// </summary>
00464 /// <param name="other">The <see cref="Rectangle"/> to check.</param>
00465 /// <returns>A boolean value indicating whether this <see cref="Rectangle"/> contains the <paramref
00466 name="other"/> <see cref="Rectangle"/>.</returns>
00467     public bool Contains(Rectangle other)
00468     {
00469         return other.X0 >= this.X0 && other.X1 <= this.X1 && other.Y0 >= this.Y0 && other.Y1 <=
00470         this.Y1;
00471
00472 /// <summary>
00473 /// Checks whether this <see cref="Rectangle"/> contains a <see cref="PointF"/>.
00474 /// </summary>
00475 /// <param name="point">The <see cref="PointF"/> to check.</param>
00476 /// <returns>A boolean value indicating whether this <see cref="Rectangle"/> contains the <paramref
00477 name="point"/>.</returns>
00478     public bool Contains(PointF point)
00479     {
00480         return point.X >= this.X0 && point.X <= this.X1 && point.Y >= this.Y0 && point.Y <=
00481         this.Y1;
00482
00483 /// <summary>
00484 /// Converts the <see cref="Rectangle"/> to a <see cref="Quad"/>.
00485 /// </summary>
00486 /// <returns>A <see cref="Quad"/> corresponding to the current <see cref="Rectangle"/>.</returns>
00487     public Quad ToQuad()
00488     {
00489         return new Quad(new PointF(X0, Y1), new PointF(X0, Y0), new PointF(X1, Y0), new PointF(X1,
00490             Y1));
00491     }
00492 }
00493
00494 /// <summary>

```

```
00492 /// Represents a rectangle using only integer numbers.
00493 /// </summary>
00494     public struct RoundedRectangle
00495     {
00496     /// <summary>
00497     /// The left coordinate of the rectangle.
00498     /// </summary>
00499         public int X0;
00500
00501     /// <summary>
00502     /// The top coordinate of the rectangle.
00503     /// </summary>
00504         public int Y0;
00505
00506     /// <summary>
00507     /// The right coordinate of the rectangle.
00508     /// </summary>
00509         public int X1;
00510
00511     /// <summary>
00512     /// The bottom coordinate of the rectangle.
00513     /// </summary>
00514         public int Y1;
00515
00516     /// <summary>
00517     /// The width of the rectangle.
00518     /// </summary>
00519         public int Width => X1 - X0;
00520
00521     /// <summary>
00522     /// The height of the rectangle.
00523     /// </summary>
00524         public int Height => Y1 - Y0;
00525
00526     /// <summary>
00527     /// Create a new <see cref="RoundedRectangle"/> from the specified coordinates.
00528     /// </summary>
00529     /// <param name="x0">The left coordinate of the rectangle.</param>
00530     /// <param name="y0">The top coordinate of the rectangle.</param>
00531     /// <param name="x1">The right coordinate of the rectangle.</param>
00532     /// <param name="y1">The bottom coordinate of the rectangle.</param>
00533         public RoundedRectangle(int x0, int y0, int x1, int y1)
00534         {
00535             X0 = x0;
00536             Y0 = y0;
00537             X1 = x1;
00538             Y1 = y1;
00539         }
00540
00541     /// <summary>
00542     /// Split the rectangle into the specified number of <see cref="RoundedRectangle"/>s.
00543     /// </summary>
00544     /// <param name="divisions">The number of rectangles in which the rectangle should be split. This
00545     /// must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than
00546     /// <paramref name="divisions"/> that satisfies this condition is used.</param>
00547     /// <returns>An array of <see cref="RoundedRectangle"/>s that when positioned properly cover the same
00548     /// area as this object.</returns>
00549         public RoundedRectangle[] Split(int divisions)
00550         {
00551             RoundedSize s = new RoundedSize(this.Width, this.Height);
00552
00553             RoundedRectangle[] splitSize = s.Split(divisions);
00554
00555             RoundedRectangle[] tbr = new RoundedRectangle[divisions];
00556
00557             for (int i = 0; i < splitSize.Length; i++)
00558             {
00559                 tbr[i] = new RoundedRectangle(this.X0 + splitSize[i].X0, this.Y0 + splitSize[i].Y0,
00560                     this.X0 + splitSize[i].X1, this.Y0 + splitSize[i].Y1);
00561             }
00562
00563     /// <summary>
00564     /// Represents a point.
00565     /// </summary>
00566         public struct PointF
00567         {
00568     /// <summary>
00569     /// The horizontal coordinate of the point.
00570     /// </summary>
00571         public float X;
00572
00573     /// <summary>
00574     /// The vertical coordinate of the point.
```

```

00575 /// </summary>
00576     public float Y;
00577
00578 /// <summary>
00579 /// Create a new <see cref="PointF"/> from the specified coordinates.
00580 /// </summary>
00581 /// <param name="x">The horizontal coordinate of the point.</param>
00582 /// <param name="y">The vertical coordinate of the point.</param>
00583     public PointF(float x, float y)
00584     {
00585         X = x;
00586         Y = y;
00587     }
00588 }
00589
00590 /// <summary>
00591 /// Represents a quadrilateral (not necessarily a rectangle).
00592 /// </summary>
00593     public struct Quad
00594     {
00595     /// <summary>
00596     /// The lower left point of the quadrilater.
00597     /// </summary>
00598     public PointF LowerLeft;
00599
00600 /// <summary>
00601 /// The upper left point of the quadrilater.
00602 /// </summary>
00603     public PointF UpperLeft;
00604
00605 /// <summary>
00606 /// The upper right point of the quadrilater.
00607 /// </summary>
00608     public PointF UpperRight;
00609
00610 /// <summary>
00611 /// The lower right point of the quadrilater.
00612 /// </summary>
00613     public PointF LowerRight;
00614
00615 /// <summary>
00616 /// Creates a new <see cref="Quad"/> from the specified points.
00617 /// </summary>
00618 /// <param name="lowerLeft">The lower left point of the quadrilater.</param>
00619 /// <param name="upperLeft">The upper left point of the quadrilater.</param>
00620 /// <param name="upperRight">The upper right point of the quadrilater.</param>
00621 /// <param name="lowerRight">The lower right point of the quadrilater.</param>
00622     public Quad(PointF lowerLeft, PointF upperLeft, PointF upperRight, PointF lowerRight)
00623     {
00624         this.LowerLeft = lowerLeft;
00625         this.UpperLeft = upperLeft;
00626         this.UpperRight = upperRight;
00627         this.LowerRight = lowerRight;
00628     }
00629
00630 /// <summary>
00631 /// Checks whether this <see cref="Quad"/> contains a <see cref="PointF"/>.
00632 /// </summary>
00633 /// <param name="point">The <see cref="PointF"/> to check.</param>
00634 /// <returns>A boolean value indicating whether this <see cref="Quad"/> contains the <paramref
00635     public bool Contains(PointF point)
00636     {
00637         return PointInTriangle(point, this.LowerLeft, this.UpperLeft, this.UpperRight) ||
00638             PointInTriangle(point, this.LowerLeft, this.UpperRight, this.LowerRight);
00639     }
00640 /// <summary>
00641 /// Checks whether a point is contained in a triangle.
00642 /// </summary>
00643 /// <param name="pt">The point to test.</param>
00644 /// <param name="A">The first vertex of the triangle.</param>
00645 /// <param name="B">The second vertex of the triangle.</param>
00646 /// <param name="C">The third vertex of the triangle.</param>
00647 /// <returns>A boolean value indicating whether the point is contained in the triangle.</returns>
00648     private static bool PointInTriangle(PointF pt, PointF A, PointF B, PointF C)
00649     {
00650         double signAB = (pt.X - B.X) * (A.Y - B.Y) - (A.X - B.X) * (pt.Y - B.Y);
00651         double signBC = (pt.X - C.X) * (B.Y - C.Y) - (B.X - C.X) * (pt.Y - C.Y);
00652         double signCA = (pt.X - A.X) * (C.Y - A.Y) - (C.X - A.X) * (pt.Y - A.Y);
00653
00654         return !((signAB < 0 || signBC < 0 || signCA < 0) && (signAB > 0 || signBC > 0 || signCA >
00655         0));
00656     }
00657 }

```

8.18 TesseractLanguage.cs

```
00001 using System;
00002 using System.Collections.Generic;
00003 using System.IO;
00004 using System.Net;
00005 using System.Reflection;
00006 using System.Text;
00007
00008 namespace MuPDFCore
00009 {
0010 /// <summary>
0011 /// Represents a language used by Tesseract OCR.
0012 /// </summary>
0013     public class TesseractLanguage
0014     {
0015     /// <summary>
0016     /// The name of the folder where the language file is located.
0017     /// </summary>
0018         public string Prefix { get; }
0019
0020     /// <summary>
0021     /// The name of the language. The Tesseract library will assume that the trained language data file
0022     /// can be found at <c>Prefix/Language.traineddata</c>.
0023     /// </summary>
0024         public string Language { get; }
0025
0026     /// <summary>
0027     /// Fast integer versions of trained models. These are models for a single language.
0028     /// </summary>
0029         public enum Fast
0030     {
0031     /// <summary>
0032     /// The Afrikaans language.
0033         Afr,
0034     /// <summary>
0035     /// The Amharic language.
0036     /// </summary>
0037         Amh,
0038     /// <summary>
0039     /// The Arabic language.
0040     /// </summary>
0041         Ara,
0042     /// <summary>
0043     /// The Assamese language.
0044     /// </summary>
0045         Asm,
0046     /// <summary>
0047     /// The Azerbaijani language.
0048     /// </summary>
0049         Aze,
0050     /// <summary>
0051     /// The Azerbaijani language (Cyrillic).
0052     /// </summary>
0053         Aze_Cyrl,
0054     /// <summary>
0055     /// The Belarusian language.
0056     /// </summary>
0057         Bel,
0058     /// <summary>
0059     /// The Bengali language.
0060     /// </summary>
0061         Ben,
0062     /// <summary>
0063     /// The Tibetan language.
0064     /// </summary>
0065         Bod,
0066     /// <summary>
0067     /// The Bosnian language.
0068     /// </summary>
0069         Bos,
0070     /// <summary>
0071     /// The Breton language.
0072     /// </summary>
0073         Bre,
0074     /// <summary>
0075     /// The Bulgarian language.
0076     /// </summary>
0077         Bul,
0078     /// <summary>
0079     /// The Catalan/Valencian language.
0080     /// </summary>
0081         Cat,
0082     /// <summary>
0083     /// The Cebuano language.
0084     /// </summary>
```

```
00085           Ceb,
00086 /// <summary>
00087 /// The Czech language.
00088 /// </summary>
00089           Ces,
00090 /// <summary>
00091 /// The Chinese (Simplified) language.
00092 /// </summary>
00093           Chi_Sim,
00094 /// <summary>
00095 /// The Chinese (Simplified) language (vertical).
00096 /// </summary>
00097           Chi_Sim_Vert,
00098 /// <summary>
00099 /// The Chinese (Traditional) language.
00100 /// </summary>
00101           Chi_Tra,
00102 /// <summary>
00103 /// The Chinese (Traditional) language (vertical).
00104 /// </summary>
00105           Chi_Tra_Vert,
00106 /// <summary>
00107 /// The Cherokee language.
00108 /// </summary>
00109           Chr,
00110 /// <summary>
00111 /// The Corsican language.
00112 /// </summary>
00113           Cos,
00114 /// <summary>
00115 /// The Welsh language.
00116 /// </summary>
00117           Cym,
00118 /// <summary>
00119 /// The Danish language.
00120 /// </summary>
00121           Dan,
00122 /// <summary>
00123 /// The German language.
00124 /// </summary>
00125           Deu,
00126 /// <summary>
00127 /// The Divehi/Dhivehi/Maldivian language.
00128 /// </summary>
00129           Div,
00130 /// <summary>
00131 /// The Dzongkha language.
00132 /// </summary>
00133           Dzo,
00134 /// <summary>
00135 /// The Greek, Modern (1453-) language.
00136 /// </summary>
00137           Ell,
00138 /// <summary>
00139 /// The English language.
00140 /// </summary>
00141           Eng,
00142 /// <summary>
00143 /// The English, Middle (1100-1500) language.
00144 /// </summary>
00145           Enm,
00146 /// <summary>
00147 /// The Esperanto language.
00148 /// </summary>
00149           Epo,
00150 /// <summary>
00151 /// A language for equations.
00152 /// </summary>
00153           Equ,
00154 /// <summary>
00155 /// The Estonian language.
00156 /// </summary>
00157           Est,
00158 /// <summary>
00159 /// The Basque language.
00160 /// </summary>
00161           Eus,
00162 /// <summary>
00163 /// The Faroese language.
00164 /// </summary>
00165           Fao,
00166 /// <summary>
00167 /// The Persian language.
00168 /// </summary>
00169           Fas,
00170 /// <summary>
00171 /// The Filipino/Pilipino language.
```

```
00172 /// </summary>
00173         Fil,
00174 /// <summary>
00175 /// The Finnish language.
00176 /// </summary>
00177         Fin,
00178 /// <summary>
00179 /// The French language.
00180 /// </summary>
00181         Fra,
00182 /// <summary>
00183 /// The German - Fraktur language.
00184 /// </summary>
00185         Frk,
00186 /// <summary>
00187 /// The French, Middle (ca.1400-1600) language.
00188 /// </summary>
00189         Frm,
00190 /// <summary>
00191 /// The Western Frisian language.
00192 /// </summary>
00193         Fry,
00194 /// <summary>
00195 /// The Gaelic/Scottish Gaelic language.
00196 /// </summary>
00197         Gla,
00198 /// <summary>
00199 /// The Irish language.
00200 /// </summary>
00201         Gle,
00202 /// <summary>
00203 /// The Galician language.
00204 /// </summary>
00205         Glg,
00206 /// <summary>
00207 /// The Greek, Ancient (to 1453) language.
00208 /// </summary>
00209         Grc,
00210 /// <summary>
00211 /// The Gujarati language.
00212 /// </summary>
00213         Guj,
00214 /// <summary>
00215 /// The Haitian/Haitian Creole language.
00216 /// </summary>
00217         Hat,
00218 /// <summary>
00219 /// The Hebrew language.
00220 /// </summary>
00221         Heb,
00222 /// <summary>
00223 /// The Hindi language.
00224 /// </summary>
00225         Hin,
00226 /// <summary>
00227 /// The Croatian language.
00228 /// </summary>
00229         Hrv,
00230 /// <summary>
00231 /// The Hungarian language.
00232 /// </summary>
00233         Hun,
00234 /// <summary>
00235 /// The Armenian language.
00236 /// </summary>
00237         Hye,
00238 /// <summary>
00239 /// The Inuktitut language.
00240 /// </summary>
00241         Iku,
00242 /// <summary>
00243 /// The Indonesian language.
00244 /// </summary>
00245         Ind,
00246 /// <summary>
00247 /// The Icelandic language.
00248 /// </summary>
00249         Isl,
00250 /// <summary>
00251 /// The Italian language.
00252 /// </summary>
00253         Ita,
00254 /// <summary>
00255 /// The Italian language (old).
00256 /// </summary>
00257         Ita_Old,
00258 /// <summary>
```

```
00259 /// The Javanese language.
00260 /// </summary>
00261         Jav,
00262 /// <summary>
00263 /// The Japanese language.
00264 /// </summary>
00265         Jpn,
00266 /// <summary>
00267 /// The Japanese language (vertical).
00268 /// </summary>
00269         Jpn_Vert,
00270 /// <summary>
00271 /// The Kannada language.
00272 /// </summary>
00273         Kan,
00274 /// <summary>
00275 /// The Georgian language.
00276 /// </summary>
00277         Kat,
00278 /// <summary>
00279 /// The Georgian language (old).
00280 /// </summary>
00281         Kat_Old,
00282 /// <summary>
00283 /// The Kazakh language.
00284 /// </summary>
00285         Kaz,
00286 /// <summary>
00287 /// The Central Khmer language.
00288 /// </summary>
00289         Khm,
00290 /// <summary>
00291 /// The Kirghiz/Kyrgyz language.
00292 /// </summary>
00293         Kir,
00294 /// <summary>
00295 /// The Northern Kurdish language.
00296 /// </summary>
00297         Kmr,
00298 /// <summary>
00299 /// The Korean language.
00300 /// </summary>
00301         Kor,
00302 /// <summary>
00303 /// The Korean language (vertical).
00304 /// </summary>
00305         Kor_Vert,
00306 /// <summary>
00307 /// The Lao language.
00308 /// </summary>
00309         Lao,
00310 /// <summary>
00311 /// The Latin language.
00312 /// </summary>
00313         Lat,
00314 /// <summary>
00315 /// The Latvian language.
00316 /// </summary>
00317         Lav,
00318 /// <summary>
00319 /// The Lithuanian language.
00320 /// </summary>
00321         Lit,
00322 /// <summary>
00323 /// The Luxembourgish/Letzeburgesch language.
00324 /// </summary>
00325         Ltz,
00326 /// <summary>
00327 /// The Malayalam language.
00328 /// </summary>
00329         Mal,
00330 /// <summary>
00331 /// The Marathi language.
00332 /// </summary>
00333         Mar,
00334 /// <summary>
00335 /// The Macedonian language.
00336 /// </summary>
00337         Mkd,
00338 /// <summary>
00339 /// The Maltese language.
00340 /// </summary>
00341         Mlt,
00342 /// <summary>
00343 /// The Mongolian language.
00344 /// </summary>
00345         Mon,
```

```
00346 /// <summary>
00347 /// The Maori language.
00348 /// </summary>
00349     Mri,
00350 /// <summary>
00351 /// The Malay language.
00352 /// </summary>
00353     Msa,
00354 /// <summary>
00355 /// The Burmese language.
00356 /// </summary>
00357     Mya,
00358 /// <summary>
00359 /// The Nepali language.
00360 /// </summary>
00361     Nep,
00362 /// <summary>
00363 /// The Dutch/Flemish language.
00364 /// </summary>
00365     Nld,
00366 /// <summary>
00367 /// The Norwegian language.
00368 /// </summary>
00369     Nor,
00370 /// <summary>
00371 /// The Occitan (post 1500) language.
00372 /// </summary>
00373     Oci,
00374 /// <summary>
00375 /// The Oriya language.
00376 /// </summary>
00377     Ori,
00378 /// <summary>
00379 /// The Orientation and script detection module.
00380 /// </summary>
00381     Osd,
00382 /// <summary>
00383 /// The Panjabi/Punjabi language.
00384 /// </summary>
00385     Pan,
00386 /// <summary>
00387 /// The Polish language.
00388 /// </summary>
00389     Pol,
00390 /// <summary>
00391 /// The Portuguese language.
00392 /// </summary>
00393     Por,
00394 /// <summary>
00395 /// The Pushto/Pashto language.
00396 /// </summary>
00397     Pus,
00398 /// <summary>
00399 /// The Quechua language.
00400 /// </summary>
00401     Que,
00402 /// <summary>
00403 /// The Romanian/Moldavian/Moldovan language.
00404 /// </summary>
00405     Ron,
00406 /// <summary>
00407 /// The Russian language.
00408 /// </summary>
00409     Rus,
00410 /// <summary>
00411 /// The Sanskrit language.
00412 /// </summary>
00413     San,
00414 /// <summary>
00415 /// The Sinhala/Sinhalese language.
00416 /// </summary>
00417     Sin,
00418 /// <summary>
00419 /// The Slovak language.
00420 /// </summary>
00421     Slk,
00422 /// <summary>
00423 /// The Slovenian language.
00424 /// </summary>
00425     Slv,
00426 /// <summary>
00427 /// The Sindhi language.
00428 /// </summary>
00429     Snd,
00430 /// <summary>
00431 /// The Spanish/Castilian language.
00432 /// </summary>
```

```
00433           Spa,
00434 /// <summary>
00435 /// The Spanish/Castilian language (old).
00436 /// </summary>
00437           Spa_Old,
00438 /// <summary>
00439 /// The Albanian language.
00440 /// </summary>
00441           Sqi,
00442 /// <summary>
00443 /// The Serbian language.
00444 /// </summary>
00445           Srp,
00446 /// <summary>
00447 /// The Serbian language (Latin).
00448 /// </summary>
00449           Srp_Latn,
00450 /// <summary>
00451 /// The Sundanese language.
00452 /// </summary>
00453           Sun,
00454 /// <summary>
00455 /// The Swahili language.
00456 /// </summary>
00457           Swa,
00458 /// <summary>
00459 /// The Swedish language.
00460 /// </summary>
00461           Swe,
00462 /// <summary>
00463 /// The Syriac language.
00464 /// </summary>
00465           Syr,
00466 /// <summary>
00467 /// The Tamil language.
00468 /// </summary>
00469           Tam,
00470 /// <summary>
00471 /// The Tatar language.
00472 /// </summary>
00473           Tat,
00474 /// <summary>
00475 /// The Telugu language.
00476 /// </summary>
00477           Tel,
00478 /// <summary>
00479 /// The Tajik language.
00480 /// </summary>
00481           Tgk,
00482 /// <summary>
00483 /// The Thai language.
00484 /// </summary>
00485           Tha,
00486 /// <summary>
00487 /// The Tigrinya language.
00488 /// </summary>
00489           Tir,
00490 /// <summary>
00491 /// The Tonga (Tonga Islands) language.
00492 /// </summary>
00493           Ton,
00494 /// <summary>
00495 /// The Turkish language.
00496 /// </summary>
00497           Tur,
00498 /// <summary>
00499 /// The Uighur/Uyghur language.
00500 /// </summary>
00501           Uig,
00502 /// <summary>
00503 /// The Ukrainian language.
00504 /// </summary>
00505           Ukr,
00506 /// <summary>
00507 /// The Urdu language.
00508 /// </summary>
00509           Urd,
00510 /// <summary>
00511 /// The Uzbek language.
00512 /// </summary>
00513           Uzb,
00514 /// <summary>
00515 /// The Uzbek language (Cyrillic).
00516 /// </summary>
00517           Uzb_Cyrl,
00518 /// <summary>
00519 /// The Vietnamese language.
```

```
00520 /// </summary>
00521             Vie,
00522 /// <summary>
00523 /// The Yiddish language.
00524 /// </summary>
00525             Yid,
00526 /// <summary>
00527 /// The Yoruba language.
00528 /// </summary>
00529             Yor
00530         }
00531
00532 /// <summary>
00533 /// Fast integer versions of trained models. These are models for a single script supporting one or
00534 /// more languages.
00535     public enum FastScripts
00536     {
00537     /// <summary>
00538     /// The Arabic script.
00539     /// </summary>
00540             Arabic,
00541     /// <summary>
00542     /// The Armenian script.
00543     /// </summary>
00544             Armenian,
00545     /// <summary>
00546     /// The Bengali script.
00547     /// </summary>
00548             Bengali,
00549     /// <summary>
00550     /// The Canadian Aboriginal script.
00551     /// </summary>
00552             Canadian_Aboriginal,
00553     /// <summary>
00554     /// The Cherokee script.
00555     /// </summary>
00556             Cherokee,
00557     /// <summary>
00558     /// The Cyrillic script.
00559     /// </summary>
00560             Cyrillic,
00561     /// <summary>
00562     /// The Devanagari script.
00563     /// </summary>
00564             Devanagari,
00565     /// <summary>
00566     /// The Ethiopic script.
00567     /// </summary>
00568             Ethiopic,
00569     /// <summary>
00570     /// The Fraktur script.
00571     /// </summary>
00572             Fraktur,
00573     /// <summary>
00574     /// The Georgian script.
00575     /// </summary>
00576             Georgian,
00577     /// <summary>
00578     /// The Greek script.
00579     /// </summary>
00580             Greek,
00581     /// <summary>
00582     /// The Gujarati script.
00583     /// </summary>
00584             Gujarati,
00585     /// <summary>
00586     /// The Gurmukhi script.
00587     /// </summary>
00588             Gurmukhi,
00589     /// <summary>
00590     /// The Han (Simplified) script.
00591     /// </summary>
00592             HanS,
00593     /// <summary>
00594     /// The Han (Simplified) script. (vertical)
00595     /// </summary>
00596             HanS_Vert,
00597     /// <summary>
00598     /// The Han (Traditional) script.
00599     /// </summary>
00600             HanT,
00601     /// <summary>
00602     /// The Han (Traditional) script. (vertical)
00603     /// </summary>
00604             HanT_Vert,
00605     /// <summary>
```

```
00606 /// The Hangul script.
00607 /// </summary>
00608         Hangul,
00609 /// <summary>
00610 /// The Hangul script. (vertical)
00611 /// </summary>
00612         Hangul_Vert,
00613 /// <summary>
00614 /// The Hebrew script.
00615 /// </summary>
00616         Hebrew,
00617 /// <summary>
00618 /// The Japanese script.
00619 /// </summary>
00620         Japanese,
00621 /// <summary>
00622 /// The Japanese script. (vertical)
00623 /// </summary>
00624         Japanese_Vert,
00625 /// <summary>
00626 /// The Kannada script.
00627 /// </summary>
00628         Kannada,
00629 /// <summary>
00630 /// The Khmer script.
00631 /// </summary>
00632         Khmer,
00633 /// <summary>
00634 /// The Lao script.
00635 /// </summary>
00636         Lao,
00637 /// <summary>
00638 /// The Latin script.
00639 /// </summary>
00640         Latin,
00641 /// <summary>
00642 /// The Malayalam script.
00643 /// </summary>
00644         Malayalam,
00645 /// <summary>
00646 /// The Myanmar script.
00647 /// </summary>
00648         Myanmar,
00649 /// <summary>
00650 /// The Oriya script.
00651 /// </summary>
00652         Oriya,
00653 /// <summary>
00654 /// The Sinhala script.
00655 /// </summary>
00656         Sinhala,
00657 /// <summary>
00658 /// The Syriac script.
00659 /// </summary>
00660         Syriac,
00661 /// <summary>
00662 /// The Tamil script.
00663 /// </summary>
00664         Tamil,
00665 /// <summary>
00666 /// The Telugu script.
00667 /// </summary>
00668         Telugu,
00669 /// <summary>
00670 /// The Thaana script.
00671 /// </summary>
00672         Thaana,
00673 /// <summary>
00674 /// The Thai script.
00675 /// </summary>
00676         Thai,
00677 /// <summary>
00678 /// The Tibetan script.
00679 /// </summary>
00680         Tibetan,
00681 /// <summary>
00682 /// The Vietnamese script.
00683 /// </summary>
00684         Vietnamese
00685     }
00686
00687 /// <summary>
00688 /// Best (most accurate) trained models. These are models for a single language.
00689 /// </summary>
00690     public enum Best
00691     {
00692 /// <summary>
```

```
00693 /// The Afrikaans language.
00694 /// </summary>
00695         Afr,
00696 /// <summary>
00697 /// The Amharic language.
00698 /// </summary>
00699         Amh,
00700 /// <summary>
00701 /// The Arabic language.
00702 /// </summary>
00703         Ara,
00704 /// <summary>
00705 /// The Assamese language.
00706 /// </summary>
00707         Asm,
00708 /// <summary>
00709 /// The Azerbaijani language.
00710 /// </summary>
00711         Aze,
00712 /// <summary>
00713 /// The Azerbaijani language (Cyrillic).
00714 /// </summary>
00715         Aze_Cyrl,
00716 /// <summary>
00717 /// The Belarusian language.
00718 /// </summary>
00719         Bel,
00720 /// <summary>
00721 /// The Bengali language.
00722 /// </summary>
00723         Ben,
00724 /// <summary>
00725 /// The Tibetan language.
00726 /// </summary>
00727         Bod,
00728 /// <summary>
00729 /// The Bosnian language.
00730 /// </summary>
00731         Bos,
00732 /// <summary>
00733 /// The Breton language.
00734 /// </summary>
00735         Bre,
00736 /// <summary>
00737 /// The Bulgarian language.
00738 /// </summary>
00739         Bul,
00740 /// <summary>
00741 /// The Catalan/Valencian language.
00742 /// </summary>
00743         Cat,
00744 /// <summary>
00745 /// The Cebuano language.
00746 /// </summary>
00747         Ceb,
00748 /// <summary>
00749 /// The Czech language.
00750 /// </summary>
00751         Ces,
00752 /// <summary>
00753 /// The Chinese (Simplified) language.
00754 /// </summary>
00755         Chi_Sim,
00756 /// <summary>
00757 /// The Chinese (Simplified) language (vertical).
00758 /// </summary>
00759         Chi_Sim_Vert,
00760 /// <summary>
00761 /// The Chinese (Traditional) language.
00762 /// </summary>
00763         Chi_Tra,
00764 /// <summary>
00765 /// The Chinese (Traditional) language (vertical).
00766 /// </summary>
00767         Chi_Tra_Vert,
00768 /// <summary>
00769 /// The Cherokee language.
00770 /// </summary>
00771         Chr,
00772 /// <summary>
00773 /// The Corsican language.
00774 /// </summary>
00775         Cos,
00776 /// <summary>
00777 /// The Welsh language.
00778 /// </summary>
00779         Cym,
```

```
00780 /// <summary>
00781 /// The Danish language.
00782 /// </summary>
00783     Dan,
00784 /// <summary>
00785 /// The German language.
00786 /// </summary>
00787     Deu,
00788 /// <summary>
00789 /// The Divehi/Dhivehi/Maldivian language.
00790 /// </summary>
00791     Div,
00792 /// <summary>
00793 /// The Dzongkha language.
00794 /// </summary>
00795     Dzo,
00796 /// <summary>
00797 /// The Greek, Modern (1453-) language.
00798 /// </summary>
00799     Ell,
00800 /// <summary>
00801 /// The English language.
00802 /// </summary>
00803     Eng,
00804 /// <summary>
00805 /// The English, Middle (1100-1500) language.
00806 /// </summary>
00807     Enm,
00808 /// <summary>
00809 /// The Esperanto language.
00810 /// </summary>
00811     Epo,
00812 /// <summary>
00813 /// The Estonian language.
00814 /// </summary>
00815     Est,
00816 /// <summary>
00817 /// The Basque language.
00818 /// </summary>
00819     Eus,
00820 /// <summary>
00821 /// The Faroese language.
00822 /// </summary>
00823     Fao,
00824 /// <summary>
00825 /// The Persian language.
00826 /// </summary>
00827     Fas,
00828 /// <summary>
00829 /// The Filipino/Pilipino language.
00830 /// </summary>
00831     Fil,
00832 /// <summary>
00833 /// The Finnish language.
00834 /// </summary>
00835     Fin,
00836 /// <summary>
00837 /// The French language.
00838 /// </summary>
00839     Fra,
00840 /// <summary>
00841 /// The German - Fraktur language.
00842 /// </summary>
00843     Frk,
00844 /// <summary>
00845 /// The French, Middle (ca.1400-1600) language.
00846 /// </summary>
00847     Frm,
00848 /// <summary>
00849 /// The Western Frisian language.
00850 /// </summary>
00851     Fry,
00852 /// <summary>
00853 /// The Gaelic/Scottish Gaelic language.
00854 /// </summary>
00855     Gla,
00856 /// <summary>
00857 /// The Irish language.
00858 /// </summary>
00859     Gle,
00860 /// <summary>
00861 /// The Galician language.
00862 /// </summary>
00863     Glg,
00864 /// <summary>
00865 /// The Greek, Ancient (to 1453) language.
00866 /// </summary>
```

```
00867           Grc,
00868 /// <summary>
00869 /// The Gujarati language.
00870 /// </summary>
00871           Guj,
00872 /// <summary>
00873 /// The Haitian/Haitian Creole language.
00874 /// </summary>
00875           Hat,
00876 /// <summary>
00877 /// The Hebrew language.
00878 /// </summary>
00879           Heb,
00880 /// <summary>
00881 /// The Hindi language.
00882 /// </summary>
00883           Hin,
00884 /// <summary>
00885 /// The Croatian language.
00886 /// </summary>
00887           Hrv,
00888 /// <summary>
00889 /// The Hungarian language.
00890 /// </summary>
00891           Hun,
00892 /// <summary>
00893 /// The Armenian language.
00894 /// </summary>
00895           Hye,
00896 /// <summary>
00897 /// The Inuktitut language.
00898 /// </summary>
00899           Iku,
00900 /// <summary>
00901 /// The Indonesian language.
00902 /// </summary>
00903           Ind,
00904 /// <summary>
00905 /// The Icelandic language.
00906 /// </summary>
00907           Isl,
00908 /// <summary>
00909 /// The Italian language.
00910 /// </summary>
00911           Ita,
00912 /// <summary>
00913 /// The Italian language (old).
00914 /// </summary>
00915           Ita_Old,
00916 /// <summary>
00917 /// The Javanese language.
00918 /// </summary>
00919           Jav,
00920 /// <summary>
00921 /// The Japanese language.
00922 /// </summary>
00923           Jpn,
00924 /// <summary>
00925 /// The Japanese language (vertical).
00926 /// </summary>
00927           Jpn_Vert,
00928 /// <summary>
00929 /// The Kannada language.
00930 /// </summary>
00931           Kan,
00932 /// <summary>
00933 /// The Georgian language.
00934 /// </summary>
00935           Kat,
00936 /// <summary>
00937 /// The Georgian language (old).
00938 /// </summary>
00939           Kat_Old,
00940 /// <summary>
00941 /// The Kazakh language.
00942 /// </summary>
00943           Kaz,
00944 /// <summary>
00945 /// The Central Khmer language.
00946 /// </summary>
00947           Khm,
00948 /// <summary>
00949 /// The Kirghiz/Kyrgyz language.
00950 /// </summary>
00951           Kir,
00952 /// <summary>
00953 /// The Northern Kurdish language.
```

```
00954 /// </summary>
00955             Kmr,
00956 /// <summary>
00957 /// The Korean language.
00958 /// </summary>
00959             Kor,
00960 /// <summary>
00961 /// The Korean language (vertical).
00962 /// </summary>
00963             Kor_Vert,
00964 /// <summary>
00965 /// The Lao language.
00966 /// </summary>
00967             Lao,
00968 /// <summary>
00969 /// The Latin language.
00970 /// </summary>
00971             Lat,
00972 /// <summary>
00973 /// The Latvian language.
00974 /// </summary>
00975             Lav,
00976 /// <summary>
00977 /// The Lithuanian language.
00978 /// </summary>
00979             Lit,
00980 /// <summary>
00981 /// The Luxembourgish/Letzeburgesch language.
00982 /// </summary>
00983             Ltz,
00984 /// <summary>
00985 /// The Malayalam language.
00986 /// </summary>
00987             Mal,
00988 /// <summary>
00989 /// The Marathi language.
00990 /// </summary>
00991             Mar,
00992 /// <summary>
00993 /// The Macedonian language.
00994 /// </summary>
00995             Mkd,
00996 /// <summary>
00997 /// The Maltese language.
00998 /// </summary>
00999             Mlt,
01000 /// <summary>
01001 /// The Mongolian language.
01002 /// </summary>
01003             Mon,
01004 /// <summary>
01005 /// The Maori language.
01006 /// </summary>
01007             Mri,
01008 /// <summary>
01009 /// The Malay language.
01010 /// </summary>
01011             Msa,
01012 /// <summary>
01013 /// The Burmese language.
01014 /// </summary>
01015             Mya,
01016 /// <summary>
01017 /// The Nepali language.
01018 /// </summary>
01019             Nep,
01020 /// <summary>
01021 /// The Dutch/Flemish language.
01022 /// </summary>
01023             Nld,
01024 /// <summary>
01025 /// The Norwegian language.
01026 /// </summary>
01027             Nor,
01028 /// <summary>
01029 /// The Occitan (post 1500) language.
01030 /// </summary>
01031             Oci,
01032 /// <summary>
01033 /// The Oriya language.
01034 /// </summary>
01035             Ori,
01036 /// <summary>
01037 /// The Orientation and script detection module.
01038 /// </summary>
01039             Osd,
01040 /// <summary>
```

```
01041 /// The Panjabi/Punjabi language.
01042 /// </summary>
01043     Pan,
01044 /// <summary>
01045 /// The Polish language.
01046 /// </summary>
01047     Pol,
01048 /// <summary>
01049 /// The Portuguese language.
01050 /// </summary>
01051     Por,
01052 /// <summary>
01053 /// The Pushto/Pashto language.
01054 /// </summary>
01055     Pus,
01056 /// <summary>
01057 /// The Quechua language.
01058 /// </summary>
01059     Que,
01060 /// <summary>
01061 /// The Romanian/Moldavian/Moldovan language.
01062 /// </summary>
01063     Ron,
01064 /// <summary>
01065 /// The Russian language.
01066 /// </summary>
01067     Rus,
01068 /// <summary>
01069 /// The Sanskrit language.
01070 /// </summary>
01071     San,
01072 /// <summary>
01073 /// The Sinhala/Sinhalese language.
01074 /// </summary>
01075     Sin,
01076 /// <summary>
01077 /// The Slovak language.
01078 /// </summary>
01079     Slk,
01080 /// <summary>
01081 /// The Slovenian language.
01082 /// </summary>
01083     Slv,
01084 /// <summary>
01085 /// The Sindhi language.
01086 /// </summary>
01087     Snd,
01088 /// <summary>
01089 /// The Spanish/Castilian language.
01090 /// </summary>
01091     Spa,
01092 /// <summary>
01093 /// The Spanish/Castilian language (old).
01094 /// </summary>
01095     Spa_Old,
01096 /// <summary>
01097 /// The Albanian language.
01098 /// </summary>
01099     Sqi,
01100 /// <summary>
01101 /// The Serbian language.
01102 /// </summary>
01103     Srp,
01104 /// <summary>
01105 /// The Serbian language (Latin).
01106 /// </summary>
01107     Srp_Latn,
01108 /// <summary>
01109 /// The Sundanese language.
01110 /// </summary>
01111     Sun,
01112 /// <summary>
01113 /// The Swahili language.
01114 /// </summary>
01115     Swa,
01116 /// <summary>
01117 /// The Swedish language.
01118 /// </summary>
01119     Swe,
01120 /// <summary>
01121 /// The Syriac language.
01122 /// </summary>
01123     Syr,
01124 /// <summary>
01125 /// The Tamil language.
01126 /// </summary>
01127     Tam,
```

```
01128 /// <summary>
01129 /// The Tatar language.
01130 /// </summary>
01131     Tat,
01132 /// <summary>
01133 /// The Telugu language.
01134 /// </summary>
01135     Tel,
01136 /// <summary>
01137 /// The Tajik language.
01138 /// </summary>
01139     Tgk,
01140 /// <summary>
01141 /// The Thai language.
01142 /// </summary>
01143     Tha,
01144 /// <summary>
01145 /// The Tigrinya language.
01146 /// </summary>
01147     Tir,
01148 /// <summary>
01149 /// The Tonga (Tonga Islands) language.
01150 /// </summary>
01151     Ton,
01152 /// <summary>
01153 /// The Turkish language.
01154 /// </summary>
01155     Tur,
01156 /// <summary>
01157 /// The Uighur/Uyghur language.
01158 /// </summary>
01159     Uig,
01160 /// <summary>
01161 /// The Ukrainian language.
01162 /// </summary>
01163     Ukr,
01164 /// <summary>
01165 /// The Urdu language.
01166 /// </summary>
01167     Urd,
01168 /// <summary>
01169 /// The Uzbek language.
01170 /// </summary>
01171     Uzb,
01172 /// <summary>
01173 /// The Uzbek language (Cyrillic).
01174 /// </summary>
01175     Uzb_Cyr1,
01176 /// <summary>
01177 /// The Vietnamese language.
01178 /// </summary>
01179     Vie,
01180 /// <summary>
01181 /// The Yiddish language.
01182 /// </summary>
01183     Yid,
01184 /// <summary>
01185 /// The Yoruba language.
01186 /// </summary>
01187     Yor
01188 }
01189
01190 /// <summary>
01191 /// Best (most accurate) trained models. These are models for a single script supporting one or more
languages.
01192 /// </summary>
01193     public enum BestScripts
01194     {
01195 /// <summary>
01196 /// The Arabic script.
01197 /// </summary>
01198     Arabic,
01199 /// <summary>
01200 /// The Armenian script.
01201 /// </summary>
01202     Armenian,
01203 /// <summary>
01204 /// The Bengali script.
01205 /// </summary>
01206     Bengali,
01207 /// <summary>
01208 /// The Canadian Aboriginal script.
01209 /// </summary>
01210     Canadian_Aboriginal,
01211 /// <summary>
01212 /// The Cherokee script.
01213 /// </summary>
```

```
01214             Cherokee,
01215     /// <summary>
01216     /// The Cyrillic script.
01217     /// </summary>
01218             Cyrillic,
01219     /// <summary>
01220     /// The Devanagari script.
01221     /// </summary>
01222             Devanagari,
01223     /// <summary>
01224     /// The Ethiopic script.
01225     /// </summary>
01226             Ethiopic,
01227     /// <summary>
01228     /// The Fraktur script.
01229     /// </summary>
01230             Fraktur,
01231     /// <summary>
01232     /// The Georgian script.
01233     /// </summary>
01234             Georgian,
01235     /// <summary>
01236     /// The Greek script.
01237     /// </summary>
01238             Greek,
01239     /// <summary>
01240     /// The Gujarati script.
01241     /// </summary>
01242             Gujarati,
01243     /// <summary>
01244     /// The Gurmukhi script.
01245     /// </summary>
01246             Gurmukhi,
01247     /// <summary>
01248     /// The Han (Simplified) script.
01249     /// </summary>
01250             HanS,
01251     /// <summary>
01252     /// The Han (Simplified) script. (vertical)
01253     /// </summary>
01254             HanS_Vert,
01255     /// <summary>
01256     /// The Han (Traditional) script.
01257     /// </summary>
01258             HanT,
01259     /// <summary>
01260     /// The Han (Traditional) script. (vertical)
01261     /// </summary>
01262             HanT_Vert,
01263     /// <summary>
01264     /// The Hangul script.
01265     /// </summary>
01266             Hangul,
01267     /// <summary>
01268     /// The Hangul script. (vertical)
01269     /// </summary>
01270             Hangul_Vert,
01271     /// <summary>
01272     /// The Hebrew script.
01273     /// </summary>
01274             Hebrew,
01275     /// <summary>
01276     /// The Japanese script.
01277     /// </summary>
01278             Japanese,
01279     /// <summary>
01280     /// The Japanese script. (vertical)
01281     /// </summary>
01282             Japanese_Vert,
01283     /// <summary>
01284     /// The Kannada script.
01285     /// </summary>
01286             Kannada,
01287     /// <summary>
01288     /// The Khmer script.
01289     /// </summary>
01290             Khmer,
01291     /// <summary>
01292     /// The Lao script.
01293     /// </summary>
01294             Lao,
01295     /// <summary>
01296     /// The Latin script.
01297     /// </summary>
01298             Latin,
01299     /// <summary>
01300     /// The Malayalam script.
```

```

01301 /// </summary>
01302         Malayalam,
01303 /// <summary>
01304 /// The Myanmar script.
01305 /// </summary>
01306         Myanmar,
01307 /// <summary>
01308 /// The Oriya script.
01309 /// </summary>
01310         Oriya,
01311 /// <summary>
01312 /// The Sinhala script.
01313 /// </summary>
01314         Sinhala,
01315 /// <summary>
01316 /// The Syriac script.
01317 /// </summary>
01318         Syriac,
01319 /// <summary>
01320 /// The Tamil script.
01321 /// </summary>
01322         Tamil,
01323 /// <summary>
01324 /// The Telugu script.
01325 /// </summary>
01326         Telugu,
01327 /// <summary>
01328 /// The Thaana script.
01329 /// </summary>
01330         Thaana,
01331 /// <summary>
01332 /// The Thai script.
01333 /// </summary>
01334         Thai,
01335 /// <summary>
01336 /// The Tibetan script.
01337 /// </summary>
01338         Tibetan,
01339 /// <summary>
01340 /// The Vietnamese script.
01341 /// </summary>
01342         Vietnamese
01343     }
01344
01345 /// <summary>
01346 /// Create a new <see cref="TesseractLanguage"/> object using the provided <paramref name="prefix"/>
01347 /// and <paramref name="language"/> name, without processing them in any way.
01348 /// </summary>
01349 /// <param name="prefix">The name of the folder where the language file is located. If this is <see
01350     public TesseractLanguage(string prefix, string language)
01351     {
01352         this.Prefix = prefix;
01353         this.Language = language;
01354     }
01355
01356 /// <summary>
01357 /// Create a new <see cref="TesseractLanguage"/> object using the specified trained model data file.
01358 /// </summary>
01359 /// <param name="fileName">The path to the trained model data file. If the file name does not end in
01360     public TesseractLanguage(string fileName)
01361     {
01362         if (fileName.EndsWith(".traineddata"))
01363         {
01364             fileName = Path.GetFullPath(fileName);
01365
01366             this.Prefix = Path.GetDirectoryName(fileName);
01367             this.Language = Path.GetFileName(fileName).Substring(0,
01368                 Path.GetFileName(fileName).Length - 12);
01369         }
01370     }
01371     else
01372     {
01373         this.Prefix = Path.GetTempPath();
01374         this.Language = Guid.NewGuid().ToString("N");
01375     }
01376
01377 }
01378
01379 private static readonly string ExecutablePath =
    Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);

```

```
01380     private static readonly string LocalCachePath =
01381         Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
01382 /// <summary>
01383 /// Create a new <see cref="TesseractLanguage"/> object using a fast integer version of a trained
01384 model for the specified language. The language file is downloaded from the
01385 <c>tesseract-ocr/tessdata_fast</c> GitHub repository. If it has already been downloaded and cached
01386 before, the downloaded file is re-used.
01387 /// </summary>
01388 /// <param name="language">The language to use for the OCR process.</param>
01389 /// <param name="useAnyCached">If this is <see langword="true"/>, if a cached trained model file is
01390 available for the specified language, it will be used even if it is a "best (most accurate)" model.
01391 Otherwise, only cached fast integer trained models will be used.</param>
01392     public TesseractLanguage(Fast language, bool useAnyCached = false)
01393     {
01394         string languageName = language.ToString().ToLower();
01395         string prefix = null;
01396
01397         if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01398             "tessdata", "fast", languageName + ".traineddata")))
01399         {
01400             prefix = Path.Combine(ExecutablePath, "tessdata", "fast");
01401         }
01402         else if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01403             "fast", languageName + ".traineddata")))
01404         {
01405             prefix = Path.Combine(ExecutablePath, "fast");
01406         }
01407         else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "fast", languageName +
01408             ".traineddata")))
01409         {
01410             prefix = Path.Combine(LocalCachePath, "tessdata", "fast");
01411         }
01412         else if (useAnyCached)
01413         {
01414             if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01415                 languageName + ".traineddata")))
01416             {
01417                 prefix = Path.Combine(ExecutablePath);
01418             }
01419             else if (!string.IsNullOrEmpty(ExecutablePath) &&
01420                 File.Exists(Path.Combine(ExecutablePath, "tessdata", "best", languageName + ".traineddata")))
01421             {
01422                 prefix = Path.Combine(ExecutablePath, "tessdata", "best");
01423             }
01424             else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "best", languageName +
01425                 ".traineddata")))
01426             {
01427                 prefix = Path.Combine(LocalCachePath, "tessdata", "best");
01428             }
01429         }
01430         if (prefix == null)
01431         {
01432             string remotePath = "https://github.com/tesseract-ocr/tessdata_fast/raw/main/" +
01433             languageName + ".traineddata";
01434
01435             string localDirectory = Path.Combine(LocalCachePath, "tessdata", "fast");
01436
01437             if (!Directory.Exists(localDirectory))
01438             {
01439                 Directory.CreateDirectory(localDirectory);
01440             }
01441
01442             using (WebClient client = new WebClient())
01443             {
01444                 client.DownloadFile(remotePath, Path.Combine(localDirectory, languageName +
01445                     ".traineddata"));
01446             }
01447
01448         /// <summary>
01449         /// Create a new <see cref="TesseractLanguage"/> object using the best (most accurate) version of the
01450 trained model for the specified language. The language file is downloaded from the
01451 <c>tesseract-ocr/tessdata_best</c> GitHub repository. If it has already been downloaded and cached
```

```

    before, the downloaded file is re-used.
01450 /// </summary>
01451 /// <param name="language">The language to use for the OCR process.</param>
01452 /// <param name="useAnyCached">If this is <see langword="true"/>, if a cached trained model file is
01453     available for the specified language, it will be used even if it is a "fast" model. Otherwise, only
01454     cached best (most accurate) trained models will be used.</param>
01455     public TesseractLanguage(Best language, bool useAnyCached = false)
01456     {
01457         string languageName = language.ToString().ToLower();
01458         string prefix = null;
01459         if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01460             "tessdata", "best", languageName + ".traineddata")))
01461         {
01462             prefix = Path.Combine(ExecutablePath, "tessdata", "best");
01463         }
01464         else if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01465             "best", languageName + ".traineddata")))
01466         {
01467             prefix = Path.Combine(ExecutablePath, "best");
01468         }
01469         else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "best", languageName +
01470             ".traineddata")))
01471         {
01472             prefix = Path.Combine(LocalCachePath, "tessdata", "best");
01473         }
01474         else if (useAnyCached)
01475         {
01476             if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01477                 languageName + ".traineddata")))
01478             {
01479                 prefix = Path.Combine(ExecutablePath);
01480             }
01481             else if (!string.IsNullOrEmpty(ExecutablePath) &&
01482                 File.Exists(Path.Combine(ExecutablePath, "tessdata", "fast", languageName + ".traineddata")))
01483             {
01484                 prefix = Path.Combine(ExecutablePath, "tessdata", "fast");
01485             }
01486             else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "fast", languageName +
01487                 ".traineddata")))
01488             {
01489                 prefix = Path.Combine(LocalCachePath, "tessdata", "fast");
01490             }
01491             if (prefix == null)
01492             {
01493                 string remotePath = "https://github.com/tesseract-ocr/tessdata_best/raw/main/" +
01494                 languageName + ".traineddata";
01495                 string localDirectory = Path.Combine(LocalCachePath, "tessdata", "best");
01496                 if (!Directory.Exists(localDirectory))
01497                 {
01498                     Directory.CreateDirectory(localDirectory);
01499                 }
01500                 using (WebClient client = new WebClient())
01501                 {
01502                     client.DownloadFile(remotePath, Path.Combine(localDirectory, languageName +
01503                         ".traineddata"));
01504                 }
01506                 prefix = localDirectory;
01507             }
01508             this.Prefix = prefix;
01509             this.Language = languageName;
01510         }
01511     }
01512 }
01513
01514 /// <summary>
01515 /// Create a new <see cref="TesseractLanguage"/> object using a fast integer version of a trained
01516     model for the specified script. The language file is downloaded from the
01517     <c>tesseract-ocr/tessdata_fast</c> GitHub repository. If it has already been downloaded and cached
01518     before, the downloaded file is re-used.
01519 /// </summary>
01520 /// <param name="script">The script to use for the OCR process.</param>
01521 /// <param name="useAnyCached">If this is <see langword="true"/>, if a cached trained model file is
01522     available for the specified script, it will be used even if it is a "best (most accurate)" model.
01523     Otherwise, only cached fast integer trained models will be used.</param>
01524     public TesseractLanguage(FastScripts script, bool useAnyCached = false)

```

```

01520         {
01521             string languageName = script.ToString().Replace("_Vert", "_vert");
01522
01523             string prefix = null;
01524
01525             if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01526             "tessdata", "fast", "script", languageName + ".traineddata")))
01527             {
01528                 prefix = Path.Combine(ExecutablePath, "tessdata", "fast", "script");
01529             }
01530             else if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01531             "fast", "script", languageName + ".traineddata")))
01532             {
01533                 prefix = Path.Combine(ExecutablePath, "fast", "script");
01534             }
01535             else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "fast", "script",
01536             languageName + ".traineddata")))
01537             {
01538                 prefix = Path.Combine(LocalCachePath, "tessdata", "fast", "script");
01539             }
01540             else if (useAnyCached)
01541             {
01542                 if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01543                 "script", languageName + ".traineddata")))
01544                 {
01545                     prefix = Path.Combine(ExecutablePath, "script");
01546                 }
01547                 else if (!string.IsNullOrEmpty(ExecutablePath) &&
01548                     File.Exists(Path.Combine(ExecutablePath, "tessdata", "best", "script", languageName +
01549                     ".traineddata")))
01550                 {
01551                     prefix = Path.Combine(ExecutablePath, "tessdata", "best", "script");
01552                 }
01553                 else if (!string.IsNullOrEmpty(ExecutablePath) &&
01554                     File.Exists(Path.Combine(ExecutablePath, "best", "script", languageName + ".traineddata")))
01555                 {
01556                     prefix = Path.Combine(LocalCachePath, "tessdata", "best", "script");
01557                 }
01558             }
01559         }
01560
01561         if (prefix == null)
01562         {
01563             string remotePath = "https://github.com/tesseract-ocr/tessdata_fast/raw/main/script/" +
01564             languageName + ".traineddata";
01565
01566             string localDirectory = Path.Combine(LocalCachePath, "tessdata", "fast", "script");
01567
01568             if (!Directory.Exists(localDirectory))
01569             {
01570                 Directory.CreateDirectory(localDirectory);
01571             }
01572
01573             using (WebClient client = new WebClient())
01574             {
01575                 client.DownloadFile(remotePath, Path.Combine(localDirectory, languageName +
01576                 ".traineddata"));
01577             }
01578
01579             prefix = localDirectory;
01580
01581             this.Prefix = prefix;
01582             this.Language = languageName;
01583         }
01584 /// <summary>
01585 /// Create a new <see cref="TesseractLanguage"/> object using the best (most accurate) version of the
01586 /// trained model for the specified script. The language file is downloaded from the
01587 /// <c>tesseract-ocr/tessdata_best</c> GitHub repository. If it has already been downloaded and cached
01588 /// before, the downloaded file is re-used.
01589 /// </summary>
01590 /// <param name="script">The script to use for the OCR process.</param>
01591 /// <param name="useAnyCached">If this is <see langword="true"/>, if a cached trained model file is
01592 /// available for the specified script, it will be used even if it is a "fast" model. Otherwise, only
01593 /// cached best (most accurate) trained models will be used.</param>
01594     public TesseractLanguage(BestScripts script, bool useAnyCached = false)
01595     {

```

```

01591     string languageName = script.ToString().Replace("_Vert", "_vert");
01592
01593     string prefix = null;
01594
01595     if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01596         "tessdata", "best", "script", languageName + ".traineddata")))
01597     {
01598         prefix = Path.Combine(ExecutablePath, "tessdata", "best", "script");
01599     }
01600     else if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01601         "best", "script", languageName + ".traineddata")))
01602     {
01603         prefix = Path.Combine(ExecutablePath, "best", "script");
01604     }
01605     else if (File.Exists(Path.Combine(LocalCachePath, "tessdata", "best", "script",
01606         languageName + ".traineddata")))
01607     {
01608         prefix = Path.Combine(LocalCachePath, "tessdata", "best", "script");
01609     }
01610     else if (useAnyCached)
01611     {
01612         if (!string.IsNullOrEmpty(ExecutablePath) && File.Exists(Path.Combine(ExecutablePath,
01613             "script", languageName + ".traineddata")))
01614         {
01615             prefix = Path.Combine(ExecutablePath, "script");
01616         }
01617         else if (!string.IsNullOrEmpty(ExecutablePath) &&
01618             File.Exists(Path.Combine(ExecutablePath, "tessdata", "fast", "script", languageName +
01619                 ".traineddata")))
01620         {
01621             prefix = Path.Combine(ExecutablePath, "tessdata", "fast", "script");
01622         }
01623         else if (!string.IsNullOrEmpty(ExecutablePath) &&
01624             File.Exists(Path.Combine(ExecutablePath, "fast", "script", languageName + ".traineddata")))
01625         {
01626             prefix = Path.Combine(LocalCachePath, "tessdata", "fast", "script");
01627         }
01628     }
01629
01630     if (prefix == null)
01631     {
01632         string remotePath = "https://github.com/tesseract-ocr/tessdata_best/raw/main/script/" +
01633             languageName + ".traineddata";
01634
01635         string localDirectory = Path.Combine(LocalCachePath, "tessdata", "best", "script");
01636
01637         if (!Directory.Exists(localDirectory))
01638         {
01639             Directory.CreateDirectory(localDirectory);
01640         }
01641
01642         using (WebClient client = new WebClient())
01643         {
01644             client.DownloadFile(remotePath, Path.Combine(localDirectory, languageName +
01645                 ".traineddata"));
01646         }
01647
01648         prefix = localDirectory;
01649     }
01650
01651     this.Prefix = prefix;
01652     this.Language = languageName;
01653 }
01654 }
```

8.19 Utils.cs

```

00001 /*
00002 MuPDFCore - A set of multiplatform .NET Core bindings for MuPDF.
00003 Copyright (C) 2020 Giorgio Bianchini
00004
00005 This program is free software: you can redistribute it and/or modify
```

```
00006 it under the terms of the GNU Affero General Public License as
00007 published by the Free Software Foundation, version 3.
00008
00009 This program is distributed in the hope that it will be useful,
00010 but WITHOUT ANY WARRANTY; without even the implied warranty of
00011 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00012 GNU Affero General Public License for more details.
00013
00014 You should have received a copy of the GNU Affero General Public License
00015 along with this program. If not, see <http://www.gnu.org/licenses/>
00016 */
00017
00018 using System;
00019
00020 namespace MuPDFCore
00021 {
00022 /// <summary>
00023 /// Contains useful methods.
00024 /// </summary>
00025     internal static class Utils
00026     {
00027     /// <summary>
00028     /// The factors by which we can divide a rectangle/size.
00029     /// </summary>
00030         public static readonly int[] AcceptableDivisors = new[] { 2, 3, 5, 7 };
00031
00032     /// <summary>
00033     /// Computes the biggest number smaller than or equal to the specified value that is factorisable
00034     /// using only the <see cref="AcceptableDivisors"/>.
00035     /// </summary>
00036     /// <param name="n">The maximum number.</param>
00037     /// <returns>A number that is &lt;= <paramref name="n"/> and is factorisable using only the <see
00038     /// cref="AcceptableDivisors"/>.</returns>
00039         public static int GetAcceptableNumber(int n)
00040         {
00041             for (int i = n; i >= 1; i--)
00042             {
00043                 if (IsAcceptableNumber(i))
00044                 {
00045                     return i;
00046                 }
00047             throw new ArgumentOutOfRangeException(nameof(n), n, "The number must be strictly higher
00048             than 0!");
00049         }
00050     /// <summary>
00051     /// Determine whether a number is factorisable using only the <see cref="AcceptableDivisors"/>.
00052     /// </summary>
00053     /// <param name="n">The number to analyse.</param>
00054     /// <returns>A boolean value indicating whether the number is factorisable using only the <see
00055     /// cref="AcceptableDivisors"/>.</returns>
00056         public static bool IsAcceptableNumber(int n)
00057         {
00058             if (n == 0)
00059             {
00060                 return false;
00061             }
00062             else if (n == 1)
00063             {
00064                 return true;
00065             }
00066             else
00067             {
00068                 for (int i = 0; i < AcceptableDivisors.Length; i++)
00069                 {
00070                     bool divided = false;
00071
00072                     while (n % AcceptableDivisors[i] == 0)
00073                     {
00074                         n /= AcceptableDivisors[i];
00075                         divided = true;
00076                     }
00077
00078                     if (divided)
00079                     {
00080                         return IsAcceptableNumber(n);
00081                     }
00082
00083                 return false;
00084             }
00085         }
00086
00087     /// <summary>
00088     /// Clear all pixels outside of a specified region.
```

```

00089 /// </summary>
00090 /// <param name="image">A pointer to the address where the pixel data is stored.</param>
00091 /// <param name="imageSize">The size in pixels of the image.</param>
00092 /// <param name="imageArea">The area represented by the image.</param>
00093 /// <param name="clipArea">The region outside which all pixels will be cleared, in image
00094 /// units.</param>
00095     public static void ClipImage(IntPtr image, RoundedSize imageSize, Rectangle imageArea,
00096     Rectangle clipArea, PixelFormats pixelFormat)
00097     {
00098         int clipLeft = Math.Max(0, (int)Math.Ceiling((clipArea.X0 - imageArea.X0) /
00099             imageArea.Width * imageSize.Width - 0.001));
00100        int clipRight = Math.Max(0, (int)Math.Floor(imageSize.Width - (imageArea.X1 - clipArea.X1) /
00101            imageArea.Width * imageSize.Width + 0.001));
00102
00103        int clipTop = Math.Max(0, (int)Math.Ceiling((clipArea.Y0 - imageArea.Y0) /
00104            imageArea.Height * imageSize.Height - 0.001));
00105        int clipBottom = Math.Max(0, (int)Math.Floor(imageSize.Height - (imageArea.Y1 -
00106            clipArea.Y1) / imageArea.Height * imageSize.Height + 0.001));
00107
00108        int pixelSize = -1;
00109        byte clearValue = 0;
00110
00111        switch (pixelFormat)
00112        {
00113            case PixelFormats.RGB:
00114            case PixelFormats.BGR:
00115                pixelSize = 3;
00116                clearValue = 255;
00117                break;
00118            case PixelFormats.RGBA:
00119            case PixelFormats.BGRA:
00120                pixelSize = 4;
00121                clearValue = 0;
00122                break;
00123        }
00124
00125        unsafe
00126        {
00127            byte* imageData = (byte*)image;
00128
00129            for (int y = 0; y < imageSize.Height; y++)
00130            {
00131                if (y < clipTop || y >= clipBottom)
00132                {
00133                    for (int x = 0; x < imageSize.Width; x++)
00134                    {
00135                        for (int i = 0; i < pixelSize; i++)
00136                        {
00137                            imageData[y * stride + x * pixelSize + i] = clearValue;
00138                        }
00139                    }
00140                }
00141                else
00142                {
00143                    for (int x = 0; x < Math.Min(clipLeft, imageSize.Width); x++)
00144                    {
00145                        for (int i = 0; i < pixelSize; i++)
00146                        {
00147                            imageData[y * stride + x * pixelSize + i] = clearValue;
00148                        }
00149
00150                    for (int x = Math.Max(0, clipRight); x < imageSize.Width; x++)
00151                    {
00152                        for (int i = 0; i < pixelSize; i++)
00153                        {
00154                            imageData[y * stride + x * pixelSize + i] = clearValue;
00155                        }
00156
00157                    }
00158                }
00159            }
00160        }
00161    }
00162
00163 /// <summary>
00164 /// Converts an image with premultiplied alpha values into an image with unpremultiplied alpha values.
00165 /// </summary>
00166 /// <param name="image">A pointer to the address where the pixel data is stored.</param>
00167 /// <param name="imageSize">The size in pixels of the image.</param>
00168     public static void UnpremultiplyAlpha(IntPtr image, RoundedSize imageSize)

```

```
00169      {
00170          int stride = imageSize.Width * 4;
00171
00172          unsafe
00173          {
00174              byte* imageData = (byte*)image;
00175
00176              for (int y = 0; y < imageSize.Height; y++)
00177              {
00178                  for (int x = 0; x < imageSize.Width; x++)
00179                  {
00180                      if (imageData[y * stride + x * 4 + 3] > 0)
00181                      {
00182                          imageData[y * stride + x * 4] = (byte)(imageData[y * stride + x * 4] * 255
00183 / imageData[y * stride + x * 4 + 3]);
00184                          imageData[y * stride + x * 4 + 1] = (byte)(imageData[y * stride + x * 4 +
00185 1] * 255 / imageData[y * stride + x * 4 + 3]);
00186                          imageData[y * stride + x * 4 + 2] = (byte)(imageData[y * stride + x * 4 +
00187 2] * 255 / imageData[y * stride + x * 4 + 3]);
00188
00189
00190
00191 }
```


Index

Abort
 MuPDFCore.MuPDFMultiThreadedPageRenderer, [114](#)

Activate
 MuPDFCore.MuPDFOptionalContentGroupConfiguration, [119](#)
 MuPDFCore.MuPDFSetOCGStateLinkDestination, [136](#)

ActivateLinks
 MuPDFCore.MuPDFRenderer.PDFRenderer, [158](#)

ActivateLinksProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, [154](#)

ActiveArea
 MuPDFCore.MuPDFLink, [108](#)

AllowDocumentModification
 MuPDFCore.PDFCreationOptions.DocumentPermissions, [50](#)

AllowDocumentRestructuring
 MuPDFCore.PDFCreationOptions.DocumentPermissions, [50](#)

Alpha
 MuPDFCore.CBZCreationOptions, [27](#)

AlternateColorSpace
 MuPDFCore.SeparationColorSpace, [181](#)

AlternativeConfigurations
 MuPDFCore.MuPDFOptionalContentGroupData, [121](#)

Annotations
 MuPDFCore.PDFCreationOptions.DocumentPermissions, [50](#)

AntiAliasing
 MuPDFCore.MuPDFContext, [67](#)

AsciiEncode
 MuPDFCore.PDFCreationOptions, [138](#)

Avalonia, [19](#)

Avalonia.Animation, [19](#)

Avalonia.Animation.RectTransition, [173](#)

Background
 MuPDFCore.MuPDFRenderer.PDFRenderer, [159](#)

BackgroundProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, [154](#)

BaseColorSpace
 MuPDFCore.IndexedMuPDFColorSpace, [55](#)

Best
 MuPDFCore.TesseractLanguage, [187](#)

BestScripts
 MuPDFCore.TesseractLanguage, [187](#)

Bounds
 MuPDFCore.MuPDFPage, [132](#)

BoxType
 MuPDFCore, [22](#)

CBZDocument
 MuPDFCore.MuPDFDocument.Create, [33, 34](#)

Chapter
 MuPDFCore.MuPDFInternalLinkDestination, [105](#)
 MuPDFCore.MuPDFOutlineItem, [128](#)

Children
 MuPDFCore.MuPDFOptionalContentGroupUIItem, [125](#)
 MuPDFCore.MuPDFOutlineItem, [129](#)

Clean
 MuPDFCore.PDFCreationOptions, [138](#)

ClearCache
 MuPDFCore.MuPDFDocument, [73](#)

ClearStore
 MuPDFCore.MuPDFContext, [66](#)

ClipToPageBounds
 MuPDFCore.MuPDFDocument, [91](#)

ColorantNames
 MuPDFCore.SeparationColorSpace, [181](#)

ColorSpace
 MuPDFCore.CBZCreationOptions, [27](#)
 MuPDFCore.MuPDFImage, [102](#)

ColorSpaceType
 MuPDFCore, [22](#)

CompressionOptions
 MuPDFCore.PDFCreationOptions, [137](#)

CompressStreams
 MuPDFCore.PDFCreationOptions, [138](#)

Contain
 MuPDFCore.MuPDFRenderer.PDFRenderer, [145](#)

Contains
 MuPDFCore.Quad, [165](#)
 MuPDFCore.Rectangle, [169](#)

Count
 MuPDFCore.MuPDFLinks, [113](#)
 MuPDFCore.MuPDFOutline, [127](#)
 MuPDFCore.MuPDFPageCollection, [134](#)

Cover
 MuPDFCore.MuPDFRenderer.PDFRenderer, [146](#)

CreateDocument
 MuPDFCore.MuPDFDocument, [73, 74](#)

Creator
 MuPDFCore.MuPDFOptionalContentGroupConfiguration, [119](#)

DefaultConfiguration

MuPDFCore.MuPDFOptionalContentGroupData, 121
Dehyphenate
 MuPDFCore.HTMLCreationOptions, 53
 MuPDFCore.TXTCreationOptions, 193
Destination
 MuPDFCore.MuPDFLink, 108
DestinationType
 MuPDFCore.MuPDFLinkDestination, 111
DisplayArea
 MuPDFCore.MuPDFRenderer.PDFRenderer, 159
DisplayAreaProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 154
DisposableIntPtr
 MuPDFCore.DisposableIntPtr, 46, 47
Dispose
 MuPDFCore.DisposableIntPtr, 47
 MuPDFCore.MuPDFContext, 66
 MuPDFCore.MuPDFDocument, 74
 MuPDFCore.MuPDFFont, 96
 MuPDFCore.MuPDFImage, 99
 MuPDFCore.MuPDFLinks, 112
 MuPDFCore.MuPDFMultiThreadedPageRenderer, 114
 MuPDFCore.MuPDFPage, 131
 MuPDFCore.MuPDFPageCollection, 133
Disposing
 MuPDFCore.LifetimeManagementException< T1, T2 >, 58
Document
 MuPDFCore.MuPDFDocument.Create, 34–36
DocumentOutputFileTypes
 MuPDFCore, 22
DocumentRestrictions
 MuPDFCore, 22
DrawLinks
 MuPDFCore.MuPDFRenderer.PDFRenderer, 159
DrawLinksProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 154
Encryption
 MuPDFCore.PDFCreationOptions, 138
EncryptionState
 MuPDFCore, 22
 MuPDFCore.MuPDFDocument, 91
EncryptionType
 MuPDFCore.PDFCreationOptions, 139
ErrorCode
 MuPDFCore.MuPDFException, 93
ExitCodes
 MuPDFCore, 23
Extraction
 MuPDFCore.PDFCreationOptions.DocumentPermissions, 50
ExtractionPermission
 MuPDFCore.PDFCreationOptions.DocumentPermissions, 49
ExtractText
 MuPDFCore.MuPDFDocument, 74, 75
ExtractTextAsync
 MuPDFCore.MuPDFDocument, 75
Fast
 MuPDFCore.TesseractLanguage, 188
FastScripts
 MuPDFCore.TesseractLanguage, 188
FormPermission
 MuPDFCore.PDFCreationOptions.DocumentPermissions, 49
Garbage
 MuPDFCore.PDFCreationOptions, 139
GarbageCollectionOption
 MuPDFCore.PDFCreationOptions, 138
GetBoundingBox
 MuPDFCore.MuPDFPage, 131
GetBytes
 MuPDFCore.MuPDFImage, 99, 100
GetEnumerator
 MuPDFCore.MuPDFLinks, 112
 MuPDFCore.MuPDFOutline, 127
 MuPDFCore.MuPDFPageCollection, 134
GetFreeTypeHandle
 MuPDFCore.MuPDFFont, 96
GetMultiThreadedRenderer
 MuPDFCore.MuPDFDocument, 76
GetProgress
 MuPDFCore.MuPDFMultiThreadedPageRenderer, 115
 MuPDFCore.MuPDFRenderer.PDFRenderer, 146
GetRenderedSize
 MuPDFCore.MuPDFDocument, 77
GetSelectedText
 MuPDFCore.MuPDFRenderer.PDFRenderer, 146
GetSpanItem
 MuPDFCore.MuPDFMultiThreadedPageRenderer, 115
GetStructuredTextPage
 MuPDFCore.MuPDFDocument, 78
GetStructuredTextPageAsync
 MuPDFCore.MuPDFDocument, 80
GetType3Handle
 MuPDFCore.MuPDFFont, 96
GraphicsAntiAliasing
 MuPDFCore.MuPDFContext, 67
GraphicsRasterizer
 MuPDFCore.CBZCreationOptions, 27
Height
 MuPDFCore.CBZCreationOptions, 28
 MuPDFCore.MuPDFImage, 102
 MuPDFCore.MuPDFInternalLinkDestination, 105
 MuPDFCore.Rectangle, 172
 MuPDFCore.RoundedRectangle, 178
 MuPDFCore.RoundedSize, 180
 MuPDFCore.Size, 184
HighlightBrush
 MuPDFCore.MuPDFRenderer.PDFRenderer, 159

HighlightBrushProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 155

HighlightedRegions
 MuPDFCore.MuPDFRenderer.PDFRenderer, 159

HighlightedRegionsProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 155

HTMLDocument
 MuPDFCore.MuPDFDocument.Create, 36, 38

ImageOrientation
 MuPDFCore.MuPDFImage, 99

IncludeAnnotations
 MuPDFCore.CBZCreationOptions, 28
 MuPDFCore.HTMLCreationOptions, 53
 MuPDFCore.PDFCreationOptions, 139
 MuPDFCore.SVGCreationOptions, 185
 MuPDFCore.TXTCreationOptions, 193

IncludeCharactersOutsideMediaBox
 MuPDFCore.HTMLCreationOptions, 53
 MuPDFCore.TXTCreationOptions, 193

InhibitSpaces
 MuPDFCore.HTMLCreationOptions, 53
 MuPDFCore.TXTCreationOptions, 193

Initialize
 MuPDFCore.MuPDFRenderer.PDFRenderer, 146–
 148

InitializeAsync
 MuPDFCore.MuPDFRenderer.PDFRenderer, 149–
 151

InputFileTypes
 MuPDFCore, 23

InternalDestinationType
 MuPDFCore.MuPDFInternalLinkDestination, 105

InternalType
 MuPDFCore.MuPDFInternalLinkDestination, 106

Intersect
 MuPDFCore.Rectangle, 170

IsBold
 MuPDFCore.MuPDFFont, 97

IsDefault
 MuPDFCore.MuPDFOptionalContentGroupConfiguration, 120

IsEnabled
 MuPDFCore.MuPDFOptionalContentGroup, 117
 MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem,
 124

IsItalic
 MuPDFCore.MuPDFFont, 97

IsLocked
 MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem,
 124

IsMonospaced
 MuPDFCore.MuPDFFont, 97

IsSerif
 MuPDFCore.MuPDFFont, 97

IsViewerInitialized
 MuPDFCore.MuPDFRenderer.PDFRenderer, 160

IsViewerInitializedProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 155

IsVisible
 MuPDFCore.MuPDFLink, 108

Items
 MuPDFCore.MuPDFOutline, 127

Label
 MuPDFCore.MuPDFOptionalContentGroupUIItem,
 126

Language
 MuPDFCore.TesseractLanguage, 191

Layout
 MuPDFCore.MuPDFDocument, 81

LayoutSinglePage
 MuPDFCore.MuPDFDocument, 81

Length
 MuPDFCore.MuPDFPageCollection, 134

LifetimeManagementException
 MuPDFCore.LifetimeManagementException<
 T1,
 T2 >, 58

Linearize
 MuPDFCore.PDFCreationOptions, 139

LinkBrush
 MuPDFCore.MuPDFRenderer.PDFRenderer, 160

LinkBrushProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 155

LinkClicked
 MuPDFCore.MuPDFRenderer.PDFRenderer, 162

LinkDestination
 MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs,
 110

LinkPen
 MuPDFCore.MuPDFRenderer.PDFRenderer, 160

LinkPenProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 156

Links
 MuPDFCore.MuPDFPage, 132

Location
 MuPDFCore.MuPDFOutlineItem, 129

LookupTable
 MuPDFCore.IndexedMuPDFColorSpace, 55

LowerLeft
 MuPDFCore.Quad, 166

LowerRight
 MuPDFCore.Quad, 166

MaxProgress
 MuPDFCore.RenderProgress.ThreadRenderProgress,
 192

Message
 MuPDFCore.LifetimeManagementException<
 T1,
 T2 >, 59

 MuPDFCore.MessageEventArgs, 60

MessageEventArgs
 MuPDFCore.MessageEventArgs, 60

MuPDFContext
 MuPDFCore.MuPDFContext, 66

MuPDFCore
 MuPDFCore, 19

 BoxType, 22

 ColorSpaceType, 22

DocumentOutputFileTypes, 22
DocumentRestrictions, 22
EncryptionState, 22
ExitCodes, 23
InputFileTypes, 23
PasswordTypes, 23
PixelFormats, 23
RasterOutputFileTypes, 23
RestrictionState, 24
MuPDFCore.BGRMuPDFColorSpace, 25
 NumBytes, 26
 Type, 26
MuPDFCore.CBZCreationOptions, 26
 Alpha, 27
 ColorSpace, 27
 GraphicsRasterizer, 27
 Height, 28
 IncludeAnnotations, 28
 RasterizerOption, 27
 RenderingColorSpace, 28
 Rotate, 28
 TextRasterizer, 28
 Width, 29
 XResolution, 29
 YResolution, 29
MuPDFCore.CMYKMuPDFColorSpace, 30
 NumBytes, 30
 Type, 30
MuPDFCore.DisposableIntPtr, 46
 DisposableIntPtr, 46, 47
 Dispose, 47
MuPDFCore.DocumentLockedException, 47
MuPDFCore.GrayscaleMuPDFColorSpace, 51
 NumBytes, 52
 Type, 52
MuPDFCore.HTMLCreationOptions, 52
 Dehyphenate, 53
 IncludeAnnotations, 53
 IncludeCharactersOutsideMediaBox, 53
 InhibitSpaces, 53
 PreserveImages, 53
 PreserveLigatures, 54
 PreserveSpans, 54
 PreserveWhitespace, 54
 UseCIDForUnknownUnicode, 54
MuPDFCore.IndexedMuPDFColorSpace, 55
 BaseColorSpace, 55
 LookupTable, 55
 NumBytes, 56
 Type, 56
MuPDFCore.LabMuPDFColorSpace, 56
 NumBytes, 57
 Type, 57
MuPDFCore.LifetimeManagementException< T1, T2
 >, 57
 Disposing, 58
 LifetimeManagementException, 58
 Message, 59
 Owner, 59
MuPDFCore.MessageEventArgs, 59
 Message, 60
 MessageEventArgs, 60
MuPDFCore.MuPDF, 61
 RedirectOutput, 61
 ResetOutput, 61
 StandardErrorMessage, 62
 StandardOutputMessage, 62
MuPDFCore.MuPDFColorSpace, 63
 Name, 64
 NumBytes, 64
 RootColorSpace, 64
 Type, 64
MuPDFCore.MuPDFContext, 65
 AntiAliasing, 67
 ClearStore, 66
 Dispose, 66
 GraphicsAntiAliasing, 67
 MuPDFContext, 66
 ShrinkStore, 66
 StoreMaxSize, 67
 StoreSize, 67
 TextAntiAliasing, 67
MuPDFCore.MuPDFDocument, 68
 ClearCache, 73
 ClipToPageBounds, 91
 CreateDocument, 73, 74
 Dispose, 74
 EncryptionState, 91
 ExtractText, 74, 75
 ExtractTextAsync, 75
 GetMultiThreadedRenderer, 76
 GetRenderedSize, 77
 GetStructuredTextPage, 78
 GetStructuredTextPageAsync, 80
 Layout, 81
 LayoutSinglePage, 81
 MuPDFDocument, 71–73
 OptionalContentGroupData, 91
 Outline, 91
 Pages, 91
 Render, 81, 82, 84, 85
 Restrictions, 92
 RestrictionState, 92
 SavelImage, 85, 86
 SavelImageAsJPEG, 87
 TryUnlock, 88
 WritelImage, 88, 89
 WritelImageAsJPEG, 90
MuPDFCore.MuPDFDocument.Create, 31
 CBZDocument, 33, 34
 Document, 34–36
 HTMLDocument, 36, 38
 PDFDocument, 38–40
 StructuredTextDocument, 40, 41
 SVGDocument, 42
 TextDocument, 42–44

XHTMLDocument, 44, 45
MuPDFCore.MuPDFException, 92
 ErrorCode, 93
MuPDFCore.MuPDFExternalLinkDestination, 93
 Type, 94
 Uri, 94
MuPDFCore.MuPDFFont, 95
 Dispose, 96
 GetFreeTypeHandle, 96
 GetType3Handle, 96
 IsBold, 97
 IsItalic, 97
 IsMonospaced, 97
 IsSerif, 97
 Name, 97
MuPDFCore.MuPDFImage, 98
 ColorSpace, 102
 Dispose, 99
 GetBytes, 99, 100
 Height, 102
 ImageOrientation, 99
 Orientation, 103
 ParentBlock, 103
 Save, 100
 SaveAsJPEG, 101
 Width, 103
 Write, 101
 WriteAsJPEG, 102
 XRes, 103
 YRes, 103
MuPDFCore.MuPDFInternalLinkDestination, 104
 Chapter, 105
 Height, 105
 InternalDestinationType, 105
 InternalType, 106
 Page, 106
 PageNumber, 106
 Type, 106
 Width, 106
 X, 107
 Y, 107
 Zoom, 107
MuPDFCore.MuPDFLink, 107
 ActiveArea, 108
 Destination, 108
 IsVisible, 108
MuPDFCore.MuPDFLinkDestination, 110
 DestinationType, 111
 Type, 111
MuPDFCore.MuPDFLinks, 112
 Count, 113
 Dispose, 112
 GetEnumerator, 112
 this[int index], 113
MuPDFCore.MuPDFMultiThreadedPageRenderer, 113
 Abort, 114
 Dispose, 114
 GetProgress, 115
 GetSpanItem, 115
 Render, 115, 116
 ThreadCount, 116
MuPDFCore.MuPDFOptionalContentGroup, 117
 Enabled, 117
 Name, 117
MuPDFCore.MuPDFOptionalContentGroupCheckbox, 118
MuPDFCore.MuPDFOptionalContentGroupConfiguration, 119
 Activate, 119
 Creator, 119
 IsDefault, 120
 Name, 120
 UI, 120
MuPDFCore.MuPDFOptionalContentGroupData, 120
 AlternativeConfigurations, 121
 DefaultConfiguration, 121
 OptionalContentGroups, 121
MuPDFCore.MuPDFOptionalContentGroupLabel, 122
MuPDFCore.MuPDFOptionalContentGroupRadioButton, 122
MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem, 123
 Enabled, 124
 Locked, 124
 Toggle, 124
MuPDFCore.MuPDFOptionalContentGroupUIItem, 125
 Children, 125
 Label, 126
MuPDFCore.MuPDFOutline, 126
 Count, 127
 GetEnumerator, 127
 Items, 127
 this[int index], 127
MuPDFCore.MuPDFOutlineItem, 128
 Chapter, 128
 Children, 129
 Location, 129
 Page, 129
 PageNumber, 129
 Title, 129
 Uri, 130
MuPDFCore.MuPDFPage, 130
 Bounds, 132
 Dispose, 131
 GetBoundingBox, 131
 Links, 132
 PageNumber, 132
MuPDFCore.MuPDFPageCollection, 133
 Count, 134
 Dispose, 133
 GetEnumerator, 134
 Length, 134
 this[int index], 134
MuPDFCore.MuPDFRenderer, 24
MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs, 109

LinkDestination, 110
MupdfLinkClickedEventArgs, 109
MuPDFCore.MuPDFRenderer.PDFRenderer, 141
ActivateLinks, 158
ActivateLinksProperty, 154
Background, 159
BackgroundProperty, 154
Contain, 145
Cover, 146
DisplayArea, 159
DisplayAreaProperty, 154
DrawLinks, 159
DrawLinksProperty, 154
GetProgress, 146
GetSelectedText, 146
HighlightBrush, 159
HighlightBrushProperty, 155
HighlightedRegions, 159
HighlightedRegionsProperty, 155
Initialize, 146–148
InitializeAsync, 149–151
IsViewerInitialized, 160
IsViewerInitializedProperty, 155
LinkBrush, 160
LinkBrushProperty, 155
LinkClicked, 162
LinkPen, 160
LinkPenProperty, 156
PageBackground, 160
PageBackgroundColor, 156
PageNumber, 160
PageNumberProperty, 156
PageSize, 161
PageSizeProperty, 156
PDFRenderer, 145
PointerEventHandlers, 145
PointerEventHandlersType, 161
PointerEventHandlerTypeProperty, 157
ReleaseResources, 152
Render, 152
RenderThreadCount, 161
RenderThreadCountProperty, 157
Search, 152
SelectAll, 153
Selection, 161
SelectionBrush, 161
SelectionBrushProperty, 157
SelectionProperty, 157
SetDisplayAreaNow, 153
Zoom, 162
ZoomEnabled, 162
ZoomEnabledProperty, 158
ZoomIncrement, 162
ZoomIncrementProperty, 158
ZoomProperty, 158
ZoomStep, 153
MuPDFCore.MuPDFRenderer/PDFRenderer.cs, 195
MuPDFCore.MuPDFRenderer/PDFRenderer.Properties.cs, 216
MuPDFCore.MuPDFRenderer/RectTransition.cs, 221
MuPDFCore.MuPDFSetOCGStateLinkDestination, 135
 Activate, 136
 Type, 136
MuPDFCore.PDFCreationOptions, 136
 AsciiEncode, 138
 Clean, 138
 CompressionOptions, 137
 CompressStreams, 138
 Encryption, 138
 EncryptionType, 139
 Garbage, 139
 GarbageCollectionOption, 138
 IncludeAnnotations, 139
 Linearize, 139
 OwnerPassword, 139
 Permissions, 140
 PrettyPrint, 140
 RegenerateId, 140
 Sanitize, 140
 UserPassword, 140
MuPDFCore.PDFCreationOptions.DocumentPermissions, 48
 AllowDocumentModification, 50
 AllowDocumentRestructuring, 50
 Annotations, 50
 Extraction, 50
 ExtractionPermission, 49
 FormPermission, 49
 Printing, 50
 PrintingPermission, 49
MuPDFCore.PointF, 163
 PointF, 163
 X, 164
 Y, 164
MuPDFCore.Quad, 164
 Contains, 165
 LowerLeft, 166
 LowerRight, 166
 Quad, 165
 UpperLeft, 166
 UpperRight, 166
MuPDFCore.Rectangle, 167
 Contains, 169
 Height, 172
 Intersect, 170
 Rectangle, 168
 Round, 170
 Split, 171
 ToQuad, 171
 Width, 172
 X0, 171
 X1, 172
 Y0, 172
 Y1, 172
MuPDFCore.RenderProgress, 173

ThreadRenderProgresses, 174
MuPDFCore.RenderProgress.ThreadRenderProgress,
 191
 MaxProgress, 192
 Progress, 192
MuPDFCore.RGBMuPDFColorSpace, 174
 NumBytes, 175
 Type, 175
MuPDFCore.RoundedRectangle, 175
 Height, 178
 RoundedRectangle, 176
 Split, 177
 Width, 178
 X0, 177
 X1, 177
 Y0, 177
 Y1, 178
MuPDFCore.RoundedSize, 178
 Height, 180
 RoundedSize, 179
 Split, 179
 Width, 180
MuPDFCore.SeparationColorSpace, 181
 AlternateColorSpace, 181
 ColorantNames, 181
 NumBytes, 182
 Type, 182
MuPDFCore.Size, 182
 Height, 184
 Size, 183
 Split, 183
 Width, 184
MuPDFCore.SVGCreationOptions, 184
 IncludeAnnotations, 185
 ReuseImages, 185
 TextOption, 185
 TextRendering, 186
MuPDFCore.TesseractLanguage, 186
 Best, 187
 BestScripts, 187
 Fast, 188
 FastScripts, 188
 Language, 191
 Prefix, 191
 TesseractLanguage, 188–190
MuPDFCore.TXTCreationOptions, 192
 Dehyphenate, 193
 IncludeAnnotations, 193
 IncludeCharactersOutsideMediaBox, 193
 InhibitSpaces, 193
 PreserveLigatures, 193
 PreserveSpans, 194
 PreserveWhitespace, 194
 UseCIDForUnknownUnicode, 194
MuPDFCore/MuPDF.cs, 222
MuPDFCore/MuPDFColorSpace.cs, 251
MuPDFCore/MuPDFContext.cs, 256
MuPDFCore/MuPDFDisplayList.cs, 259
 MuPDFCore/MuPDFDocument.Create.cs, 260
 MuPDFCore/MuPDFDocument.cs, 281
 MuPDFCore/MuPDFFont.cs, 304
 MuPDFCore/MuPDFImage.cs, 306
 MuPDFCore/MuPDFLinks.cs, 313
 MuPDFCore/MuPDFMultiThreadedPageRenderer.cs,
 318
 MuPDFCore/MuPDFOptionalContentGroupConfiguration.cs,
 326
 MuPDFCore/MuPDFOutline.cs, 332
 MuPDFCore/MuPDFPage.cs, 335
 MuPDFCore/Rectangles.cs, 339
 MuPDFCore/TesseractLanguage.cs, 347
 MuPDFCore/Utils.cs, 366
 MuPDFDocument
 MuPDFCore.MuPDFDocument, 71–73
 MupdfLinkClickedEventArgs
 MuPDFCore.MuPDFRenderer.MupdfLinkClickedEventArgs,
 109
Name
 MuPDFCore.MuPDFColorSpace, 64
 MuPDFCore.MuPDFFont, 97
 MuPDFCore.MuPDFOptionalContentGroup, 117
 MuPDFCore.MuPDFOptionalContentGroupConfiguration,
 120
NumBytes
 MuPDFCore.BGRMuPDFColorSpace, 26
 MuPDFCore.CMYKMuPDFColorSpace, 30
 MuPDFCore.GrayscaleMuPDFColorSpace, 52
 MuPDFCore.IndexedMuPDFColorSpace, 56
 MuPDFCore.LabMuPDFColorSpace, 57
 MuPDFCore.MuPDFColorSpace, 64
 MuPDFCore.RGBMuPDFColorSpace, 175
 MuPDFCore.SeparationColorSpace, 182
OptionalContentGroupData
 MuPDFCore.MuPDFDocument, 91
OptionalContentGroups
 MuPDFCore.MuPDFOptionalContentGroupData,
 121
Orientation
 MuPDFCore.MuPDFImage, 103
Outline
 MuPDFCore.MuPDFDocument, 91
Owner
 MuPDFCore.LifetimeManagementException< T1,
 T2 >, 59
OwnerPassword
 MuPDFCore.PDFCreationOptions, 139
Page
 MuPDFCore.MuPDFInternalLinkDestination, 106
 MuPDFCore.MuPDFOutlineItem, 129
PageBackground
 MuPDFCore.MuPDFRenderer.PDFRenderer, 160
PageBackgroundProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 156
PageNumber

MuPDFCore.MuPDFInternalLinkDestination, 106
 MuPDFCore.MuPDFOutlineItem, 129
 MuPDFCore.MuPDFPage, 132
 MuPDFCore.MuPDFRenderer.PDFRenderer, 160
PageNumberProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 156
Pages
 MuPDFCore.MuPDFDocument, 91
PageSize
 MuPDFCore.MuPDFRenderer.PDFRenderer, 161
PageSizeProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 156
ParentBlock
 MuPDFCore.MuPDFImage, 103
PasswordTypes
 MuPDFCore, 23
PDFDocument
 MuPDFCore.MuPDFDocument.Create, 38–40
PDFRenderer
 MuPDFCore.MuPDFRenderer.PDFRenderer, 145
Permissions
 MuPDFCore.PDFCreationOptions, 140
PixelFormats
 MuPDFCore, 23
PointerEventHandlers
 MuPDFCore.MuPDFRenderer.PDFRenderer, 145
PointerEventHandlersType
 MuPDFCore.MuPDFRenderer.PDFRenderer, 161
PointerEventHandlerTypeProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 157
PointF
 MuPDFCore.PointF, 163
Prefix
 MuPDFCore.TesseractLanguage, 191
PreserveImages
 MuPDFCore.HTMLCreationOptions, 53
PreserveLigatures
 MuPDFCore.HTMLCreationOptions, 54
 MuPDFCore.TXTCreationOptions, 193
PreserveSpans
 MuPDFCore.HTMLCreationOptions, 54
 MuPDFCore.TXTCreationOptions, 194
PreserveWhitespace
 MuPDFCore.HTMLCreationOptions, 54
 MuPDFCore.TXTCreationOptions, 194
PrettyPrint
 MuPDFCore.PDFCreationOptions, 140
Printing
 MuPDFCore.PDFCreationOptions.DocumentPermissions, 50
PrintingPermission
 MuPDFCore.PDFCreationOptions.DocumentPermissions, 49
Progress
 MuPDFCore.RenderProgress.ThreadRenderProgress, 192
Quad
 MuPDFCore.Quad, 165
RasterizerOption
 MuPDFCore.CBZCreationOptions, 27
RasterOutputFileTypes
 MuPDFCore, 23
Rectangle
 MuPDFCore.Rectangle, 168
RedirectOutput
 MuPDFCore.MuPDF, 61
RegenerateId
 MuPDFCore.PDFCreationOptions, 140
ReleaseResources
 MuPDFCore.MuPDFRenderer.PDFRenderer, 152
Render
 MuPDFCore.MuPDFDocument, 81, 82, 84, 85
 MuPDFCore.MuPDFMultiThreadedPageRenderer, 115, 116
 MuPDFCore.MuPDFRenderer.PDFRenderer, 152
RenderingColorSpace
 MuPDFCore.CBZCreationOptions, 28
RenderThreadCount
 MuPDFCore.MuPDFRenderer.PDFRenderer, 161
RenderThreadCountProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 157
ResetOutput
 MuPDFCore.MuPDF, 61
Restrictions
 MuPDFCore.MuPDFDocument, 92
RestrictionState
 MuPDFCore, 24
 MuPDFCore.MuPDFDocument, 92
ReuseImages
 MuPDFCore.SVGCreationOptions, 185
RootColorSpace
 MuPDFCore.MuPDFColorSpace, 64
Rotate
 MuPDFCore.CBZCreationOptions, 28
Round
 MuPDFCore.Rectangle, 170
RoundedRectangle
 MuPDFCore.RoundedRectangle, 176
RoundedSize
 MuPDFCore.RoundedSize, 179
Sanitize
 MuPDFCore.PDFCreationOptions, 140
Save
 MuPDFCore.MuPDFImage, 100
SaveAsJPEG
 MuPDFCore.MuPDFImage, 101
SaveImage
 MuPDFCore.MuPDFDocument, 85, 86
SaveImageAsJPEG
 MuPDFCore.MuPDFDocument, 87
Search
 MuPDFCore.MuPDFRenderer.PDFRenderer, 152
SelectAll
 MuPDFCore.MuPDFRenderer.PDFRenderer, 153
Selection
 MuPDFCore.MuPDFRenderer.PDFRenderer, 161

SelectionBrush
 MuPDFCore.MuPDFRenderer.PDFRenderer, 161

SelectionBrushProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 157

SelectionProperty
 MuPDFCore.MuPDFRenderer.PDFRenderer, 157

SetDisplayAreaNow
 MuPDFCore.MuPDFRenderer.PDFRenderer, 153

ShrinkStore
 MuPDFCore.MuPDFContext, 66

Size
 MuPDFCore.Size, 183

Split
 MuPDFCore.Rectangle, 171
 MuPDFCore.RoundedRectangle, 177
 MuPDFCore.RoundedSize, 179
 MuPDFCore.Size, 183

StandardErrorMessage
 MuPDFCore.MuPDF, 62

StandardOutputMessage
 MuPDFCore.MuPDF, 62

StoreMaxSize
 MuPDFCore.MuPDFContext, 67

StoreSize
 MuPDFCore.MuPDFContext, 67

StructuredTextDocument
 MuPDFCore.MuPDFDocument.Create, 40, 41

SVGDocument
 MuPDFCore.MuPDFDocument.Create, 42

TesseractLanguage
 MuPDFCore.TesseractLanguage, 188–190

TextAntiAliasing
 MuPDFCore.MuPDFContext, 67

TextDocument
 MuPDFCore.MuPDFDocument.Create, 42–44

TextOption
 MuPDFCore.SVGCreationOptions, 185

TextRasterizer
 MuPDFCore.CBZCreationOptions, 28

TextRendering
 MuPDFCore.SVGCreationOptions, 186

this[int index]
 MuPDFCore.MuPDFLinks, 113
 MuPDFCore.MuPDFOutline, 127
 MuPDFCore.MuPDFPageCollection, 134

ThreadCount
 MuPDFCore.MuPDFMultiThreadedPageRenderer, 116

ThreadRenderProgresses
 MuPDFCore.RenderProgress, 174

Title
 MuPDFCore.MuPDFOutlineItem, 129

Toggle
 MuPDFCore.MuPDFOptionalContentGroupSelectableUIItem, 124

ToQuad
 MuPDFCore.Rectangle, 171

TryUnlock

Type
 MuPDFCore.BGRMuPDFColorSpace, 26
 MuPDFCore.CMYKMuPDFColorSpace, 30
 MuPDFCore.GrayscaleMuPDFColorSpace, 52
 MuPDFCore.IndexedMuPDFColorSpace, 56
 MuPDFCore.LabMuPDFColorSpace, 57
 MuPDFCore.MuPDFColorSpace, 64
 MuPDFCore.MuPDFExternalLinkDestination, 94
 MuPDFCore.MuPDFInternalLinkDestination, 106
 MuPDFCore.MuPDFLinkDestination, 111
 MuPDFCore.MuPDFSetOCGStateLinkDestination, 136
 MuPDFCore.RGBMuPDFColorSpace, 175
 MuPDFCore.SeparationColorSpace, 182

UI

UpperLeft
 MuPDFCore.Quad, 166

UpperRight
 MuPDFCore.Quad, 166

Uri
 MuPDFCore.MuPDFExternalLinkDestination, 94
 MuPDFCore.MuPDFOutlineItem, 130

UseCIDForUnknownUnicode
 MuPDFCore.HTMLCreationOptions, 54
 MuPDFCore.TXTCreationOptions, 194

UserPassword
 MuPDFCore.PDFCreationOptions, 140

Width
 MuPDFCore.CBZCreationOptions, 29
 MuPDFCore.MuPDFImage, 103
 MuPDFCore.MuPDFInternalLinkDestination, 106
 MuPDFCore.Rectangle, 172
 MuPDFCore.RoundedRectangle, 178
 MuPDFCore.RoundedSize, 180
 MuPDFCore.Size, 184

Write
 MuPDFCore.MuPDFImage, 101

WriteAsJPEG
 MuPDFCore.MuPDFImage, 102

WriteImage
 MuPDFCore.MuPDFDocument, 88, 89

WriteImageAsJPEG
 MuPDFCore.MuPDFDocument, 90

X
 MuPDFCore.MuPDFInternalLinkDestination, 107
 MuPDFCore.PointF, 164

X0
 MuPDFCore.Rectangle, 171
 MuPDFCore.RoundedRectangle, 177

X1
 MuPDFCore.Rectangle, 172
 MuPDFCore.RoundedRectangle, 177

XHTMLDocument

MuPDFCore.MuPDFDocument.Create, [44, 45](#)

XRes
MuPDFCore.MuPDFImage, [103](#)

XResolution
MuPDFCore.CBZCreationOptions, [29](#)

Y
MuPDFCore.MuPDFInternalLinkDestination, [107](#)
MuPDFCore.PointF, [164](#)

Y0
MuPDFCore.Rectangle, [172](#)
MuPDFCore.RoundedRectangle, [177](#)

Y1
MuPDFCore.Rectangle, [172](#)
MuPDFCore.RoundedRectangle, [178](#)

YRes
MuPDFCore.MuPDFImage, [103](#)

YResolution
MuPDFCore.CBZCreationOptions, [29](#)

Zoom
MuPDFCore.MuPDFInternalLinkDestination, [107](#)
MuPDFCore.MuPDFRenderer.PDFRenderer, [162](#)

ZoomEnabled
MuPDFCore.MuPDFRenderer.PDFRenderer, [162](#)

ZoomEnabledProperty
MuPDFCore.MuPDFRenderer.PDFRenderer, [158](#)

ZoomIncrement
MuPDFCore.MuPDFRenderer.PDFRenderer, [162](#)

ZoomIncrementProperty
MuPDFCore.MuPDFRenderer.PDFRenderer, [158](#)

ZoomProperty
MuPDFCore.MuPDFRenderer.PDFRenderer, [158](#)

ZoomStep
MuPDFCore.MuPDFRenderer.PDFRenderer, [153](#)